# CA SOLVE:Access™ Session Management

## Security Guide

r5

This documentation and any related computer software help programs (hereinafter referred to as the "Documentation") are for your informational purposes only and are subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be used or disclosed by you except as may be permitted in a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

# CA Technologies Product References

This document references the following CA Technologies products:

- CA ACF2™ for z/OS (CA ACF2 for z/OS)
- CA NetMaster® Network Management for SNA (CA NetMaster NM for SNA)
- CA SOLVE:Access™ Session Management (CA SOLVE:Access)
- CA SOLVE:FTS
- CA SOLVE:InfoMaster™
- CA SOLVE:NetMail™
- CA Top Secret® for z/OS (CA Top Secret for z/OS)

# Contact CA Technologies

**Contact Technical Support**

For your convenience, CA Technologies provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

**Provide Feedback**

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA Technologies product documentation, complete our short customer survey, which is available on the CA Support website at http://ca.com/docs.

# Contents

## Chapter 1: Understanding Security 13

## Chapter 2: Using the NMSAF Security Solution 19

## Chapter 3: Working with UAMS 27

## Chapter 4: Working with an External Security Exit for User IDs 41

## Chapter 5: Administering Security 55

## Chapter 6: Implementing Security Exits 57

## Appendix A: SXCTL Parameters 67

## Appendix B: Structured Fields 79

## Appendix C: User ID Security Exit Support 141

## Appendix D: Data Set Authorization Exits Support 187

# Chapter 1: Understanding Security

Your products require a sophisticated security system, because:

- Each product has many features.
- The features often have varied security requirements.
- The products have many users.

Setting up a security regime is an important part of the implementation of each product. This guide helps you to make the best choice of security system, and guides you through the necessary implementation steps.

**Note:** This guide contains descriptive text and procedures about options and products that you may not be licensed for or have not enabled. Inclusion of the descriptions of these options and products in this guide in no way implies that you are licensed for these options or products.

This section contains the following topics:

## Security System Options

Your product region can use internal or external security systems, or a combination of the two. The options available are:

- UAMS—the internal security interface
- Partial security exit
- The NMSAF solution
- Full security exit

## UAMS

The internal security configuration that your product region can use is UAMS (User ID Access Maintenance System). In this configuration, all information about authorized users, including user ID, password, name, and privileges, is stored in a VSAM data set.

Because this is an internal security interface, your product region does not interface to any external security system or product.

## Partial Security Exit

The partial security exit configuration that your product region can use is a hybrid. The UAMS data set still exists, but an exit is used in conjunction with it. The exit interfaces to an external security system, and performs (at least) user ID and password validation. In this case, the UAMS data set still contains user information and privileges. Passwords are not stored in the UAMS data set.

This is the most useful configuration. As described in the following section, a comprehensive hybrid security solution, NMSAF, is provided that uses a partial security exit.

## NMSAF Solution

A comprehensive security solution is shipped with your product. This solution is known as NMSAF.

The NMSAF solution is built around a partial security exit. It uses the UAMS data set to store specific information for your product region, and uses whatever security product is installed (through the IBM-defined SAF interfaces) to perform user validation and password checking.

NMSAF uses its own parameter file (SXCTL) to provide flexible implementation.

## Full Security Exit

The full security exit configuration that your product region can use consists of just an exit. In this case, the exit performs all user authentication. It also supplies all user attributes and privilege information. There is no UAMS data set.

## Recommended Options

From the options available, which is the best choice for you? This choice depends on:

- Whether you are running your product region for production or testing purposes
- How specific or stringent your security requirements are

We recommend the following options to suit different requirements:

- For a product region used for production, we recommend you use the NMSAF solution. You can implement this solution with minimal work. The solution provides a comprehensive set of facilities that make administration of security for your product region straightforward.
- For a product region used for testing, you can use NMSAF. However, in some cases, simply using UAMS is sufficient (for example, if only one or two people use the product region).
- If you have specific or stringent requirements, consider writing your own partial or full security exit, which is not a trivial task.

**Note:** The NMSAF solution has flexibility and can meet all or most of your needs.

**More information:**

Using the NMSAF Security Solution (see page 19)

## Signon Access

Signon access to a region is controlled by one or more of the following:

- The User ID Access Maintenance Subsystem (UAMS)
- The NMSAF security solution
- An external security package that performs some or all the security functions through a full or partial security exit.

## UAMS

UAMS is a database of user details and access authority levels used by your product. You can maintain all security details (including user passwords) in UAMS, or you can replace UAMS, either partially or fully, with an external security package.

You can either define each user's user ID separately or add users with the same security requirements by using a UAMS group.

## NMSAF Security Solution

The NMSAF security solution (see page 19) is based on the partial security exit facility. It does not replace UAMS but works in conjunction with it.

## User ID Security Exits

If your organization has an external security package, such as CA ACF2, CA Top Secret, or IBM RACF, access to that package is provided through one of the following types of exit:

- **Partial security exit**—password and logon access maintenance is controlled by the external security package while UAMS stores the user access authorities.
- **Full security exit**—all security functions are maintained and stored by your external security package.

**More information:**

User ID Security Exit Support (see page 141)

# Additional Security Options

Your product provides additional security options in the following areas:

- File access from NCL—can be restricted by using the NCL authorization exit, NCLEX01
- INMC link activation—can be verified for authority by using the INMC security exit
- The ALLOCATE command and CA SOLVE:FTS—can be secured by using the data set access authorization exit
- The data set services interface ($DSCALL)—can be secured by using the data set services authorization exit

**More information:**

Implementing Security Exits (see page 57)

# Chapter 2: Using the NMSAF Security Solution

The NMSAF security solution is an integrated security management system for users of your regions.

The NMSAF security solution is based on the partial security exit facility and works with UAMS. The solution provides the following features:

- A complete security solution for your region, using whatever external security system is in use
- A sensible balance between what is stored in the external security system for your users and resources, and what is maintained on UAMS
- Control and customizing options that allow for flexible implementation

The NMSAF security solution minimizes duplication between external security definitions and UAMS. By using the NMSAF security solution, it is possible to eliminate almost all maintenance issues associated with using a UAMS data set.

This section contains the following topics:

## Components of NMSAF

The NMSAF security solution consists of the following components:

- **UAMS**—NMSAF uses the UAMS file to store user records that contain the access authorization details for a user ID.
- **Partial security exit**—NMSAF uses a partial security exit to interface with your external security package for password checking. Passwords are not stored in UAMS.
- **Modeling**—You can use the modeling facility to reduce the number of users that must be defined to your product region (by using UAMS). When you use modeling, a set of model users is defined. Each model user definition is used to define the privileges that a specific type of user has.

  The NMSAF parameter file defines a list of resource names and associated model names. When a user that is not defined to UAMS logs on, this list is searched and each resource name is tested to see whether the user has READ access. The model user ID of the first one that matches is then used as the basis of a new user ID definition.
- **SXCTL parameter file**—The SXCTL file is the control file used by NMSAF. You can use the SXCTL file as supplied or you can tailor it to your requirements by using parameters.
- **Additional security exits**—NMDSNCHK and NMDSSCHK can work with NMSAF. Several other exits are supplied.

**More information:**

Understanding Security (see page 13)

## User Groups and Modeling with NMSAF

With user groups, you can classify users by the type of functions that they have access to. User groups are defined in the UAMS file.

### Benefits of Using Groups and Modeling

User groups simplify the definition of user records—a user is allocated to a group, inheriting all of its access authorizations.

Using both groups and modeling provides the following combination of benefits:

- When your region models a user, a copy of the model user ID record is produced.
- By containing only the group name in this record, you ensure that the UAMS records (created as users are modeled) contain only unique user-specific information (such as user ID, user name, and telephone number).
- To change the profiles of all users in a group, you need only change the group entry in UAMS.
- To move a user from one group to another, you need only update the user's UAMS record to point to the correct group name.
- When a user logs on to your product region for the first time, that user is tested against the listed resource names. When a resource that the user has permission to access is found, the associated model definition is used to create the user's UAMS record. The user is prompted to supply specific information, such as name and telephone number. However, everything else is taken from the model user for the appropriate group.

## How You Implement NMSAF

To implement the NMSAF security solution with user groups and modeling enabled, you perform the following tasks:

1. Defining Your User Groups
2. Modeling Your User Groups
3. Enabling NMSAF (see page 24)

**Note:** If your UAMS data set is empty, you log on with the INSTALL user ID before you perform these tasks. For more information, see *Installation Guide*.

## Define Your User Groups

User groups simplify the definition of user records—a user is allocated to a group, inheriting all of its access authorizations.

**To define your user groups**

1. Determine a set of logical user groups.

   Each group can have specific authority needs in your region.

   **Note:** You can manually define users with significant privileges using UAMS (see page 27).

2. For each group, create a UAMS GROUP user definition with the appropriate set of privileges.

## Model Your User Groups

You can implement modeling by defining model user IDs that inherit the attributes of defined group IDs. You create resources for the user groups in your external security package. You then associate these model user IDs with the resources.

**To model your user groups**

1. For each defined GROUP user, create a single model user ID.

2. Using your external security package, create resource names for each defined group. These names must use the same resource class name as the SXCTL RCLASS setting (default FACILITY); for example:

   - SOLVE.ADMIN, for an administrator
   - SOLVE.OPER, for a system operator
   - SOLVE.NOPER, for a network operator
   - SOLVE.MON, for a monitor user

   **Notes:**

   - These resource names are generic. If you have several product regions and you want users to have different profiles on each, you could use the ACB name or domain name of each region as part of the name (for example, SOLVE.ADMIN.NM01).
   - If you use a different class name, define the class to the security system.

3. Issue commands to define and activate the resources in your external security system. Give PERMIT privileges with READ access to the appropriate resource to all users that access your region.

4. Set up the SXCTL file with the following statements:

   ```
   MODEL LIST
   MODELGROUP   resource.name.1   model1
   MODELGROUP   resource.name.2   model2
   MODELGROUP   resource.name.3   model3
   MODELGROUP   resource.name.4   model4
   ```

   **Note:** List the resource names in the order that you want them to be tested.

   If you want to allow a generic logon for any other users, add an additional line:

   ```
   MODELGROUP   *   dfltmodel
   ```

**More information:**

External Security Definitions for Modeled Users (see page 225)
User ID Modeling (see page 46)

## Enable NMSAF

**To enable the NMSAF security solution**

1. Set the JCL parameter SEC to SEC=NMSAF.

   You can set SEC=NMSAF either during initial implementation of your product, or later.

   When you set SEC=NMSAF, you activate the NMSAF partial security exit, and so enable the NMSAF security solution.

   If you require other components of the NMSAF security solution, activate them separately.

   Use the procedures described in Customizing the SXCTL Parameter File (see page 25) and Additional Security Exits (see page 26).

   **Note:** For a full description of the JCL parameter SEC, see the *Reference Guide*.

   **Note:** If your Security product is CA Top Secret, create a region control definition for signon (see page 25).

2. Restart your region.

   The security exit picks up the definitions.

### Implement Signon and Signoff with CA Top Secret

External security includes security for signon and signoff. The CA Top Secret security administrator must create a region control ACID, FACILITY, and started task definition for the online STC (SOLVE).

**To create this definition**

1. Create a region control ACID using the following commands:

   ```
   TSS CRE(netmacid) NAME('region_acid SOLVE') DEPT(netmdept) PASS(NOPW,0)
   FAC(STC,SOLVE) MASTFAC(SOLVE) NOVOLCHK NORESCHK NOLCFCHK NODSNCHK NOSUBCHK
   ```

2. Create a SOLVE FACILITY by placing the following statements into the CA Top Secret startup parameter file member:

   ```
   FAC(user15=NAME=SOLVE)
   FAC(SOLVE=NOABEND,ASUBM)
   FAC(SOLVE=INSTDATA,KEY=8,LCFCMD,LOCKTIME=0,NOLUMSG)
   FAC(SOLVE=MULTIUSER,PGM=NM0,NORNDPW,RES,SIGN(M))
   FAC(SOLVE=SHRPRF,NOSTMSG,NOTSOC,WARNPW,NOXDEF)
   ```

3. Define the SOLVE STC to the CA Top Secret STC table using the following command:

   ```
   TSS ADD(STC)  PROCNAME(SOLVE)  ACID(netmacid)
   ```

4. For any region control ACID to be used to sign on, authorize it to the SOLVE FACILITY using the following command:

   ```
   TSS ADD(user1) IBMFAC(SOLVE)
   ```

## SXCTL Parameter File

If SEC=NMSAF is in effect, there is an optional parameter file, accessed through DD SXCTL. You can use this file to customize the NMSAF facility to suit the security needs of your installation.

A sample SXCTL file is in the PARMLIB member SXPARMS. For more information, see the comments in this sample file.

**Note:** If the SXCTL file is not allocated, then default settings are used for all parameters.

**More information:**

SXCTL Parameters (see page 67)

# Additional Security Exits

The following security exits are available to use with NMSAF:

- NMSECDSN (for the NMDSNCHK exit type)
- NMSECDSS (for the NMDSSCHK exit type)

The NMSECDSN exit works with the NMSAF security solution to provide user level security authorization for CA SOLVE:FTS functions and the ALLOCATE command.

The NMSECDSS exit works with the NMSAF security solution to provide user level security authorization for data set services functions.

**More information:**

Data Set Allocation Authority (see page 64)
Security for Data Set Services (see page 65)
Data Set Authorization Exits Support (see page 187)

# Chapter 3: Working with UAMS

This section contains the following topics:

## UAMS

UAMS is designed to provide a fully self-contained system for user security. The subsystem allows you to define user IDs for each user of your regions. User IDs provide logon and password checking and can be added, deleted, or updated.

You can use UAMS to define:

- Each user ID separately
- A group ID to be used as a model for each user that requires similar access and authority

UAMS by itself is independent of any external security system. For example, passwords stored in UAMS are not synchronized with RACF.

Multiple regions can share a single UAMS data set.

# Background User IDs

The following UAMS background user ID definitions (where *nnnn* is the domain ID) are defined when a region is started for the first time:

***nnnn*BLOG**

Defines the logger background user.

***nnnn*BMON**

Defines the monitor background user.

***nnnn*BSVR**

Defines the server background user.

***nnnn*BSYS**

Defines the system background user.

***nnnn*LOGP**

Defines the log procedure background user.

# Users to the Region

Different types of user ID can be defined to your region:

- Individual user
- Model
- System console
- Background environments
  - Background process
  - System procedure

The following sections describe how to define each of these user IDs.

## Define a User ID

Before anyone can access your region, they must be defined as an authorized user. A one- to eight-character user identifier (the user ID), together with a password, is used to associate an individual to the privileges and authorities allocated them. The password can be maintained by UAMS or by an external security system.

In addition to identifying an individual user of the system, the user ID also defines the following information about a user:

- Statistical information—personal information such as name, location, telephone number, as well as user ID expiry date and start times of last session.
- Control information—identifies the functions the user is allowed to perform and the features they can access when logged on to the system. This information determines, for example, whether a user is allowed to act as an OCS operator and if so, their authority level .

**To define an individual user ID for a new user**

1. Enter **/UAMS** at the command prompt.

   The UAMS : Primary Menu is displayed.

2. Type **A** at the Select Option prompt.
3. Type in the User Name.
4. Type **USER** in the Definition Type field and press Enter.

   The first of several panels of user ID definition details is displayed. These describe a user’s access to various features.

   **Note:** You can associate a user with a group definition (for example, one of the groups generated when your region was installed). To associate a user with a group definition, simply enter the group name in the Group ID field on the first panel. Doing this means that the user inherits the privileges set in the group definition.

5. Type the required information on each following panel, scrolling forward (F8) to review the next panel.
6. Press F3 (File) to save the new user ID definition when all the panels have been reviewed and the required attributes specified.

   You are returned to the UAMS : Primary Menu.

**Note:** If the user ID you are defining is similar to another ID, you can save time by copying an existing user ID to a new user you wish to add. Do this by selecting option L (List) to list the existing users, and then option C (Copy) to copy the user definition. UAMS retrieves the details for that user ID and then enters Add mode. You must now enter the new user ID name and change any fields required, before filing the new definition under the new name.

## Maintain User IDs

You can obtain a selection list of the user IDs defined to your system. The list allows you to browse, force a password, update, delete, or copy any of the listed user IDs. If you are not authorized for UAMS maintenance functions, only the browse function is allowed.

The following information is displayed about each user ID:

- Name
- Location
- Telephone number
- Type (Group or User definition)
- Access privileges

### To maintain an existing user ID

1. Type **L** at the Select Option prompt on the UAMS : Primary Menu.
2. To obtain a partial listing, enter a prefix in the User field to list only those user IDs beginning with that prefix.
3. To display the command authority and access privileges information, press F11 (Right) twice.

## Force a Password Change for a User ID

If a user forgets their password, you can allocate a new password using the Force User's Password Change facility.

**To allocate a new password using the Force User's Password Change facility**

1. Enter **/UAMS** at the command prompt.

   The UAMS : Primary Menu is displayed.

2. Type **F** at the Select Option prompt on the UAMS : Primary Menu, change the User field to the user ID you want to change the password for, and press Enter.

   The UAMS : User Details panel for the specified User ID is displayed.

3. Type the new password in the New Password field, and press Enter.

   The following message displays:

   ```
   PASSWORD VERIFICATION COMPLETE
   ```

4. Press F3 (File) to save the changes.

   When the user next logs on to the region, they are prompted to change the forced password. If a security exit is provided for password processing, this option can be suppressed by the exit.

**Note:** This function is not available from the special Install User ID.

# Model User ID

Many installations have numerous users who access only one or two functions. Defining and maintaining system access for individual users requires considerable administrative effort. To minimize this effort, you can set up a model user ID so that users can automatically log on and register.

The SYSPARMS MODLUSER command specifies this model user ID.

The command has the following format:

```
SYSPARMS MODLUSER={ userid | NONE }
```

**Note:** For a detailed description of the MODLUSER operand, see the *Reference Guide*.

## How It Works

If no security exit is in place and a model user ID is defined, a user who tries to log on with an undefined user ID, but using the password of the model user ID, causes the following to occur:

- The model user ID definition is read.
- A new user ID is created using the user ID specified during the logon request with the attributes of the model user ID.
- The new user ID is saved and flagged as a new user.
- The user ID creation is recorded on the log.
- Logon is allowed to proceed.
- The new user is prompted to change their password and fill in personal details (name, telephone, location, and so on) before the logon is complete.

**Note:** Creating new user IDs from model definitions is not suitable when defining high authority user IDs.

If no security exit is in place, and no model user ID has been defined, logon attempts from user IDs that are not defined are rejected.

If a model user ID has been specified in the SYSPARMS MODLUSER operand but has not been defined to UAMS, logon requests by undefined users will fail as if no model was defined.

If a security exit is in place, model user IDs work differently (see page 46).

## System Console User ID

The system console needs a special type of user ID because of the following characteristics:

- The console only receives messages.
- The console has no full-screen capabilities.
- User logon is not possible.

The system console user ID can be defined in the same manner as any other user ID, however, only fields that are applicable to message receipt are valid. For example, by defining the system console user ID as a monitor status OCS operator with PPO authority and an appropriate command authority, you can profile the system console as an OCS operator console in the same manner as any other user ID.

The following sections describe how the system console user ID is created.

## How a Region Assigns the System Console User ID in z/OS

The z/OS environment supports named consoles and extended MCS consoles, RACF signon, and security for consoles.

The console user ID can be reproduced in UAMS so that a user who has limited authority cannot circumvent that authority by going to a console and issuing a MODIFY command.

**Default OPER Environment**

A default CONSOLE environment allows messages to be delivered to the operator. These messages are then delivered using the routing and descriptor codes set by the SYSPARMS ROUTCDE and DESC operands. This environment is built after INIT has finished.

The default terminal name for the system console environment is CONSOLE. The user ID for the console is automatically assigned using the following process:

1. The value of the SYSPARMS SYSCONUI operand is examined.
2. If no value is defined, it looks at the default—*pppp*OPER, where *pppp* is the system user prefix as defined in the NMSUP region JCL parameter.
3. If there is no definition for *pppp*OPER, the region assigns .DFLTOP as the user ID.

If .DFLTOP is used because no other value is defined, problems with ROF routing to other regions can result.

**Note:** For more information about the SYSPARMS DESC, ROUTCDE, and SYSCONUI operands, see the *Reference Guide*.

**Actual Console Environments**

A system console environment is signed on the first time that a command is sent from the console to the system (for example, MODIFY).

The terminal name used is one of the following:

- The MVS console name if:
  - SYSPARMS SYSCONNM=ALL is in effect
  - It is an extended MVS console
- CONS#*nn* or CONS#*nnn*

The user ID depends on the values of the following SYSPARMS operands:

- SYSCONUI—sets the basic user ID. If this operand is not specified, it defaults to *pppp*OPER.
- SYSCONSO—determines the search order for user IDs when a user attempts to sign on to the console.

If SYSPARMS SYSCONXU=NO is in effect, the user ID is determined as shown in the following table, with one exception. The extended MCS consoles use the SYSCONUI user ID value as the logical user ID because a user ID cannot be derived from the console ID.

When an attempt to log on to the system console is made, the system tries three times to assign a user ID to the system console. The outcome depends on the value of the SYSCONSO operand, as shown in the following table:

| | **Attempt 1** | **Attempt 2** | **Attempt 3** |
|---|---|---|---|
| SYSCONSO= DEFAULT | User ID is set to *ppppCNxx* or *ppppCxxx* | User ID is set to the value of SYSCONUI | User ID is set to .DFLTOP |
| SYSCONSO= NO | User ID is set to the value of SYSCONUI | User ID is set to .DFLTOP | |
| SYSCONSO= REQUIRED | User ID is set to *ppppCNxx* or *ppppCxxx* | | |

If SYSCONXU=YES is in effect, the values of the SYSPARMS SYSCONSO and SYSCONUI operands determine the user ID:

- SYSCONUI—sets the basic user ID. If not specified, it defaults to *pppp*OPER.
- SYSCONSO—determines the search order for user IDs when a user attempts to sign on to the console

When an attempt to log on to the system console is made, the system tries three times to assign a user ID to the system console. The outcome depends on the value of the SYSCONSO operand. The following table describes this process:

| | **Attempt 1** | **Attempt 2** | **Attempt 3** |
|---|---|---|---|
| SYSCONSO= DEFAULT | User ID is set to the RACF user ID signed on at the console | User ID is set to the value of SYSCONUI | User ID is set to .DFLTOP |
| SYSCONSO= NO | User ID is set to the value of SYSCONUI | User ID is set to .DFLTOP | |
| SYSCONSO= REQUIRED | User ID is set to the RACF user ID signed on at the console | | |

The console is signed on by trying each attempt in turn until one succeeds.

**Note:** For detailed information about the SYSPARMS SYSCONUI, SYSCONSO, SYSCONXU, and SYSCONNM operands, see the *Reference Guide*.

When a RACF user ID is signed on at the system console, there are two special cases, as follows:

- If a user is not signed on at the master console, RACF uses an internal name of *BYPASS*. This defaults to .MASTOP in this case.

  If .MASTOP is encountered, the signon always succeeds, with system assigned defaults.

- If the user is not signed on for other consoles, an internal user ID of .NOTSIGN is used.

  If .NOTSIGN is encountered, the signon of that user ID fails, leading to a try of the next user ID, and so on. If it is the last, the signon fails completely.

### Receiving Command Replies on the System Console

All commands entered at the system console in an MVS system are treated as private to that console. The results of the commands entered are returned only to that console.

### ROF and System Consoles

System consoles can establish ROF sessions with remote domains.

If a user ID is defined to UAMS for a specific console, then a corresponding user ID must be defined on the remote domains.

If no specific console user ID is defined and the console is operating with the same privileges as the console user, the console user ID can establish ROF connections to other domains without those domains requiring specific user ID definitions. The console user ID uses the ROF attributes of the console user instead.

### MSGPROC and System Console User IDs

The user ID environment for a system console can have a standard MSGPROC associated with it. MSGPROC processing is activated automatically during the console's signon.

### Unsolicited Output to the System Console

To have the system console receive unsolicited messages, for example, Monitor class messages, direct them to a console user ID. The default system routing codes then determine which physical consoles receive the messages.

## Background Environment User IDs

Two types of background environments exist:

- Background processes, which include:
  - BMON—background monitor
  - BLOG—background logger
  - BSYS—background system process
- System procedures, which execute in special system-level environments logically signed on before the procedure starts (for examples, LOGPROC)

## Initialization User IDs

Each background environment is assigned a special user ID by the system at initialization. These user IDs are formed by using the system user prefix as defined in the NMSUP initialization parameter. For example, if your system has a system user prefix of NM01:

- The background environment user ID is defined as NM01BMON.
- The LOGPROC system procedure user ID is defined as NM01LOGP.

**Note:** Background environments cannot be canceled.

To see the names of these processes on your own system, enter a SHOW USERS command to list background environment users.

## Initialization Privileges

When the system initializes, the background environment users are logically signed on. If a UAMS user ID is defined for a background environment, the attributes and privileges for it are determined from the user ID definition. If no user ID is defined, the system assigns the background environment with the following privileges:

- Time zone of the system
- Maximum command authority
- OCS authority

**Note:** UAMS background user ID definitions are created automatically when your product region starts for the first time.

## ROF and Background Environments

Background environments can have ROF sessions with connected domains. Background environments must have their user IDs defined to all the remote domains that they will log on to.

## MSGPROC and Background Environment User IDs

Background environment user IDs can have standard MSGPROCs associated with them. To associate a MSGPROC with a background environment user ID, update the user ID in UAMS to include MSGPROC.

## How You Retrieve User ID Information Using NCL

The NCL verb &SECCALL allows you to access the attributes of any nominated user ID defined to UAMS. The &SECCALL GET statement retrieves a nominated user ID and presents the requested user ID information to an NCL procedure.

You can access all the information about a specified user ID or you can specify the structured fields (see page 79) that identify particular fields within the user ID. The following syntax is used to obtain this information:

```
&SECCALL GET USERID=userid FIELDS=(nnnn,...,nnnn)
```

Each field that is retrieved from the user ID is given a default name generated as follows:

&SEC*nnnn*

where &SEC is the default prefix, and *nnnn* is a four-digit number corresponding to one of the structured field keys used to identify fields within the user ID.

### Example:

The structured field key used to identify the user ID name is 0010. When this field is retrieved, it has a name of &SEC0010.

**Note:** Only those user ID privileges relevant to the configuration of the system in which the &SECCALL GET statement is executed can be retrieved. The &SECCALL GET statement operates in the same manner if a full security exit is implemented that supports the relevant calls generated by these statements.

**More information:**

Structured Fields (see page 79)

# Chapter 4: Working with an External Security Exit for User IDs

This section contains the following topics:

## User ID Security Exits

A user ID security exit can be used to provide partial or full security processing:

- A *partial* security exit *supplements* UAMS by replacing the password checking part of UAMS with external security system validation of the user ID and password. Other user profile information is still maintained on UAMS, although the exit has the option of supplementing or overriding this information.
- A *full* security exit *replaces* UAMS. All security and user profile information must be supplied by the exit.

### Source Code for Sample Exits

Source code for sample exits is distributed with your product. These include a SAF partial exit, CA ACF2 full exits, and RACF full exits. These are supplied as-is, to show how an exit should be written.

**Note:** Using any form of security exit can have ramifications on some products; for example, on system user IDs. These issues are explained in more detail in the following sections.

## Partial Security Exits

If your region operates with a partial security exit, then UAMS password checking functions are disabled. Instead, the exit is called to validate a user ID and password. Typically, this is done with a call to the external security system (for example, RACF).

The exit can also supply overriding or additional user profile attributes. The exit can also control modeling, whereby users can be dynamically created in the UAMS data set the first time that they use this product.

## PARTSAF Partial Security Exit

Your product includes a standard partial security exit that uses SAF to communicate with your external security package. If the JCL parameter SEC=PARTSAF is coded, then your region operates with this partial security exit. This exit performs straightforward processing; for example, for logons, it requires the user to be defined to the external security system and validates the password. If a model name is set (by using the SYSPARMS MODLUSER command), and the user is not known to the region, then the user is defined to the region and the nominated model user ID is used to build the user's profile in UAMS.

## Full Security Exits

If your region operates with a full security exit, then no UAMS data set is used. The security exit must perform all required security functions. Specifically, it must provide all information about user authority on the region.

## Considerations When Using a Security Exit

Some products might not operate correctly if a security exit is in use. This is because many products make use of system users to perform work. A system user is an internal user, automatically logged on. System users do not correspond to any real user. The internal logon occurs even if the security exit says that the user is not defined. These users can log on to other regions. However, when this occurs, the target region's security exit attempts to validate the user.

Thus, if you are using a security exit, these user IDs (that log on to other regions) must be defined to the external security system. However, no specific password is required, because the validation call simply checks that the user is known.

If you have many regions that interconnect, then, by using the NMSUP JCL parameter, you can reduce the number of unique user IDs that must be defined this way. Set the value of NMSUP for all the regions to the same value (for example, NETM). The system user IDs in each region will then have the same names (for example, NETMBSYS, NETMBLOG, NETMAOMP). By default, the prefix is the value of the NMDID parameter.

# Functions Performed by a User ID Security Exit

The following security functions can be implemented by using the user ID security exit:

- Control access to your system
- Perform logon verification
- Allow users to change their password
- Retrieve user ID definitions
- Update user ID definitions
- Add additional security exit functions
- List user ID definitions
- Add user ID definitions
- Delete user ID definitions

Each of these functions is described below.

**More information:**

User ID Security Exit Support (see page 141)

## Access to Your Region

Access to your region is controlled by providing user logon security. Your exit performs the following functionality:

- Verify whether the user ID is authorized to access the region.
- Confirm the attributes and privileges that the user has when logged on to the region.

The exit is called to accept logon attempts from the following sources:

- Native terminal logons
- TSO/TSS interface logons from the External Interface Package (EIP)
- ROF logons
- Operating system console logons
- System environment logons
- APPC user region logons
- Model user ID logons

When a user attempts to log on, the user ID and password (if applicable) is passed to the exit for confirmation that the logon can proceed.

When a partial security exit is installed, and the user ID is defined on the UAMS data set, the user ID attributes are also passed to the exit.

### External Interface Package (EIP) Logons

EIP logons can originate from TSO or BCI. These calls do not need to supply a password.

## ROF Logons

ROF logon requests might originate from domains that implement differing levels of security. In order to assist the exit to reach a decision with ROF logon requests, the following information is provided to the exit:

- The INMC link name of the domain from which the request came
- The domain ID (if available) of the originating domain
- A flag byte indicating whether the domain ID is present
- A flag byte indicating whether the ROF request originated from a domain that is different from the one identified by the INMC link name

The SIGNON command allows a password to be specified. The exit can check whether a password was specified for a ROF logon, and refuse the logon if no password was specified.

## Operating System Console Logon

When an operating system console is first attached, a logon request is made. The exit can return user ID information applicable to the profile required for the particular console or it can indicate that the user ID is unknown.

**More information:**

System Console User ID (see page 33)

## System Environment Logons

A logon call is generated for each system environment during system or procedure initialization. No passwords are associated with system environment logons.

If the exit rejects the logon by setting return code 24, default values are assigned for the system environment logon.

**More information:**

Working with UAMS (see page 27)

## APPC User Region Logon

When an APPC transaction is defined with conversation level security, an APPC user region logon is performed to validate and sign on the partner transaction program region.

Depending on the type of APPC logon being performed, a password may or may not be provided. If one is provided, it should be checked. If none is provided, it indicates that the region is being started from a known, valid environment. In this case the user ID should be validated, with no password check.

## User ID Modeling

Using a partial security exit allows more flexibility with model user IDs. The following scenarios can be specified in the exit:

- The SYSPARMS MODLUSER command can provide a system default model name that is supplied to the exit to create a new user ID.
- The SYSPARMS MODLUSER model user ID can be used, or an alternative model can be nominated to define a new user ID.
- A model user ID can be nominated to override the existing attributes of an already defined user ID.
- The exit can modify any or all of the individual attributes of that user ID by supplying a group of structured fields.

In this manner, the exit has complete control over both known and unknown users wishing to log on to your region.

When a model is specified, the exit changes the value of the 8-character user ID name addressed by word 3 of the Logon Request parameter list, to the user ID name for the nominated model user ID. You can define many model user IDs, for example, ADMIN, NETOP, SYSOP, or SYSPROG.

The exit should validate both user ID and password before allowing an automatically modeled user creation. Otherwise, simple mistakes from mistyped user IDs might generate spurious user ID definitions on the UAMS data set.

**Note:** Model users take precedence over changes to the user ID. If the exit specifies a model user ID but the model is itself not defined on UAMS the logon attempt is rejected with a user not known condition. This is the case regardless of whether the user is defined on UAMS or not, and the logon fails.

### Password Status of Modeled User IDs

By default, all users defined using the model user ID are new users, and all new users have to change their password when they first log on. This is not convenient if users have the same password for all systems.

To avoid this situation, the partial security exit must indicate that the logon password is correct and that the user ID is to be created and treated as an existing user ID. This means that a password change is not enforced before allowing the logon.

If an unknown user logs on and is automatically given a new user ID based on a model designated by the exit, the new user ID is created on UAMS:

- If the exit sets return code 0, then the new user ID is created but the user is not asked to change their password.
- If the exit sets return code 4, then the user is requested to change their password since the exit is indicating the password for that user ID has expired.
- If the exit sets return code 8, then the new user ID is created but the user is prompted to change their password since they are classified as a new user.

## Logon Verification

When a user is logged on to the region, there are times when their password is verified. Password verification is needed in the following circumstances:

- When a user enters their password to resume use of a locked terminal
- When a user attempts to alter the MAI-FS details that use the &USERPW function
- For any NCL procedure using &SECCALL CHECK

Logon verification functionality is supplied by the &SECCALL CHECK verb. When there is no external security package being used, UAMS handles &SECCALL CHECK.

### Model User ID Verification

Consider the following points when model user IDs are subject to logon verification:

- Consistent logic should be used in the coding of user logon and logon verification calls, and they should use the same model user IDs.
- If a user ID is not defined to UAMS, then the exit can be coded to either reject the user ID or specify a user ID to be used as a model.
- If the SYSPARMS MODLUSR operand is specified, all calls that pass the user ID to the exit also pass the model user ID.

### APPC Link Verification

Verification is also required when an APPC link is started and the link is defined with PASSWORD=EXIT.

The exit is designed to use SAF APPCLU class. When a request for an APPC link is made, session partners are passed to the exit as an entity defined to the SAF security system in the following form:

```
NETID.LU1.LU2
```

where LU1 is the requesting system and LU2 is the target system. To remove the necessity to define NETID.LU1.LU2 and NETID.LU2.LU1 with the same session key, you should specify the security exit to swap LU1 and LU2.

## Password Change

To help ensure the security of your system, users must be able to change their password in the following circumstances:

- At any time by using the PASSWORD command or **/CHGPWD**
- When they log on and their password has expired

## Retrieval of User ID Information

When a user is logged on to your system, they can enter commands that require the retrieval of their user ID attributes. Provide this functionality in the following circumstances:

- When a SHOW OCS command is issued—name and location is obtained for all users currently using OCS
- When a PROFILE INITCMD command is issued— retrieves a copy of the user ID definition to update their OCS profile
- For any NCL procedure using &SECCALL GET
- When a user ID definition is updated

The required user ID information is returned as a set of structured fields.

If you have a partial security system, your region retrieves the definition of the required user ID from UAMS. The region then presents the definition to the exit as a set of structured fields for inspection or modification before completing the request.

If you have a full security system, the security exit must provide all the structured fields for the user definition.

**More information:**

Processing of Structured Fields (see page 54)

## Updates to User IDs

User IDs must be updated when information pertaining to the user has changed. Provide this functionality in the following circumstances:

- When a user ID requires a change in its privileges
- When a user wants to update their user details
- For any NCL procedure using &SECCALL UPDATE
- For any NCL procedure using &SECCALL CHANGE with the FIELDS or DETAILS operands

Updating a user ID requires the ability to retrieve user ID information.

If you do not want to override any user attributes from UAMS, then return the parameter list unchanged and set the return code to zero. This consideration applies to partial security exits only.

The exit does not need to support the ability to update user IDs, but if it does not, some other method of changing a user’s INITCMD and user details must be available.

## Additional Security Functions

You can add your own functions to the security exit; for example, to obtain statistics about the exit’s performance. The &SECCALL EXIT statement provides this functionality by allowing you to communicate between the security exit and NCL procedures. Communication is performed by passing the contents of nominated variables to your security exit.

On return to the NCL procedure the contents of the variables, passed as parameter areas, are placed in individual NCL variables named &1, &2, &3, and so on. A parameter area that was assigned a zero data length by the exit sets a null value.

**Note:** The exit can return only as many variables to the NCL procedure as were nominated on the original &SECCALL EXIT statement.

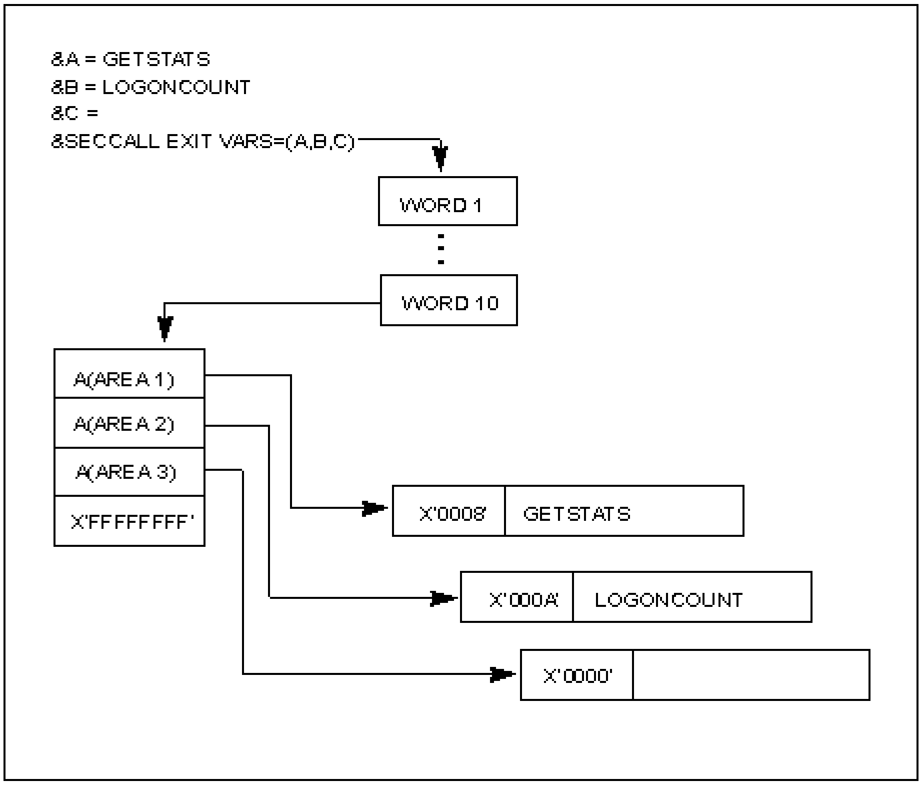
### Example:

To obtain statistics about the number of logons performed by the security exit, set the following variables:

```
&A=GETSTATS
&B=LOGONCOUNT
&C=
```

These variables are passed to the exit as parameters by the &SECCALL EXIT VARS=(A,B,C) statement. The exit modifies the contents of these parameters, setting the data length in the first 2 bytes of the parameter area. The maximum length of the parameter area is 256 bytes.

The following diagram shows how the exit deals with an &SECCALL EXIT call.



When the exit returns the variables to the NCL procedure, the values of the parameter areas are placed in &1, &2, and &3.

The exit can also set a return code that is passed back to the NCL procedure as the value of the system variable &RETCODE. In addition, the exit can supply a message which is returned to the procedure in &SYSMSG.

&SECCALL EXIT is available for use in any NCL procedure, but is valid only in systems which are configured with a security exit.

**Note:** For detailed information about &SECCALL EXIT, see the *Network Control Language Reference Guide*.

## Listing of User IDs (Full Security Exit Only)

You can obtain a list of all the user IDs defined to your system. To build this list, you retrieve sequential user IDs. The OPT=KGT or OPT=KLT option of the &SECCALL GET NCL statement performs this functionality. This statement retrieves user ID definition details for an NCL procedure of the next (KGT) or previous (KLT) user ID defined following a nominated user ID name.

If you use a partial security exit, this call is not made to the exit. A sequential get is performed on the UAMS data set and then a specific return user ID information call is passed to the security exit.

## Addition of User IDs

You must be able to add a new user ID definition so that a new user can access the region. This functionality is provided by the following:

- The UAMS Add function
- Any NCL procedure using &SECCALL ADD USERID

If you use a partial security exit and you do not want to override any of the attributes, set the return code to zero and return the parameter list unchanged to add the user definition to the UAMS data set.

If you use a full security exit, this function need not be implemented. Your external security package performs this function.

## Deletion of User IDs

You must be able to delete a user ID definition. This functionality is provided by:

- The UAMS Delete function
- Any NCL procedure using &SECCALL DELETE USERID

If you use a full security exit, this function need not be implemented. Your external security package performs this function.

## Processing of Structured Fields

If the user logging on is defined to UAMS, their user ID definition privileges are presented to the exit as a list of structured fields (see page 79). Word 10 of the logon request parameter list addresses this list.

By translating the definition of a known user ID into structured fields and presenting them to the exit at logon time, the exit can inspect or modify the attributes and privileges of the user.

The exit cannot add structured fields to the list provided on the call but the following actions can be taken:

- Any structured field passed to the exit can be modified.
- A structured field can be logically deleted from the list by clearing its address pointer from the list.
- The exit can provide a complete replacement set of structured fields by replacing the address pointer in Word 10 of the parameter list.

# Chapter 5: Administering Security

This section contains the following topics:

## Command Authority Levels

Product commands and VTAM commands that your product can execute are each assigned a default authority level. These levels are within the range of 0 to 255.

Users of your product region are also allocated a command authority level. This level is in their user ID definition. The authority level set corresponds to the authority level for the commands they are authorized to issue.

### Change Command Authority Levels

To allocate a different command authority to a particular command, use the SYSPARMS CMDAUTH operand.

**Example:**

To allocate a command authority of 5 to the CANCEL command, enter the following command:

```
SYSPARMS CMDAUTH=(CANCEL,5)
```

### Disable Commands

To disable a command, allocate an authority level higher than the level available to any user.

## Replace Commands with NCL Procedures

To replace native commands with an NCL procedure of the same name, use the CMDREPLS parameter group in Customizer.

### Example: Replace the CANCEL Command

**To run an NCL procedure in place of the CANCEL command**

1. Enter **/PARMS** at the command prompt.

   The Customizer : Parameter Groups panel is displayed.

2. Enter **U** next to the CMDREPLS parameter group.

   The Customizer : Parameter Group panel for the CMDREPLS parameter group is displayed.

3. Type **CANCEL** in the next available position in the Replaced Command ID field.

4. Press one of the following function keys:

   - F3 (File) to file the changes—the changes are not applied to the current region, but are applied when the region is restarted.
   - F6 (Action) to apply the changes to this region—the changes are not saved.

# Chapter 6: Implementing Security Exits

Implementation of security exits can provide additional security for your systems.

Your product provides some special-purpose exits that you can use to perform security checks. The following exits are supported:

- NCL authorization exit
- INMC security exit
- Data set access authorization exit
- Data set services authorization exit

This section contains the following topics:

# Security for File Access

The NCL authorization exit, NCLEX01, controls access to the following:

- User databases (UDBs)
- SQL databases in a DB2 environment
- Writing SMF records (z/OS systems)
- Network databases (NDBs)

Access can be restricted to certain levels, or can be denied altogether. For example, the level of access to a database can be used to restrict users to read only or update without delete.

The exit is invoked automatically the first time an NCL procedure attempts to open a database, using the &FILE OPEN, &EDB, or &NDBOPEN statement, or write a record using the &SMFWRITE statement. A return code from the exit sets the level of access permitted, which is made available to the NCL procedure.

**Note:** If an exit controls the writing of SMF records in z/OS systems, it must provide clearance for a user to use the &SMFWRITE verb.

You can also define other uses for the NCL exit by tailoring the supplied NCLEX01 or writing your own.

**More information:**

How the NCL Authorization Exit Works (see page 59)

## How the NCL Authorization Exit Works

To activate the NCL authorization exit, specify the name of the load module to invoke in the SYSPARMS NCLEX01 operand. To ensure that the exit is always activated during system initialization, place the SYSPARMS command in the INIT initialization member. If no exit is required, specify SYSPARMS NCLEX01=NO. If necessary the name of the exit can be changed online and a new module invoked.

**Note:** For information about the SYSPARM NCLEX01 operand, see the *Reference Guide*.

**Note:** If you are using a security product, such as RACF, to control VSAM file access, ensure that your product region itself is authorized for file access.

The exit executes within a subtask and can therefore issue a WAIT or SVC that causes suspension of the task without affecting the primary task.

The exit is passed a parameter list that provides information about the request. Write the exit as a reentrant facility so that multiple concurrent requests are possible.

For performance reasons, calls to NCLEX01 are not generated to the high-usage system files MODS and $PSPOOL.

### Errors in the Exit

If an ABEND occurs within the exit, any current request is rejected as though access had been denied. Although this can impact the requesting NCL procedure, other processing is not affected. Subsequent requests to the exit continue without impact.

### Parameters Passed to the Exit

When the exit is invoked, it is passed a communication area that provides information about the request and the requestor. The $NMNCEX1 macro, as supplied in the macro library, describes this area. The macro is in the *dsnpref*.*pvpref*.CC2DMAC data set.

***dsnpref***

Is your site-specific data set name prefix.

***pvpref***

Is your product version prefix.

## Preload the NCL Authorization Exit

You can load a single copy of the exit into the region in advance to avoid the overhead of the loading process.

To preload the NCL authorization exit, specify the name of the exit in the LOAD MODULE operand.

If the exit is link edited with the RENT (reentrant) attribute, this single copy is shared concurrently among multiple requestors.

**Important!** We recommended that you make the NCL exit module reentrant. Failure to do so causes significant additional disk activity and resource consumption in a busy system, and can impact overall performance.

## Additional Checking in the NCL Authorization Exit

You can provide additional checking for access through the NCL authorization exit. This additional information is specified on the &FILE OPEN statement.

**Example:**

To have the exit ask for the user's password to confirm access to a particular UDB, you code the password information in the exit as follows:

```
&FILE OPEN ID=MYFILE DATA=&MYPASSWD
&FILE OPEN ID=MYFILE FORMAT=UNMAPPED DATA=&MYPASSWD
```

The exit interrogates the password and set an appropriate return code to control the subsequent processing in the NCL procedure.

A maximum of 50 characters of data can be passed. This data is available to the exit in the NEXUDATA field. The length of the data passed is set in the NEXUDLEN field. No validation is performed on passed data.

## Information Sharing between NCL Authorization and External Security Exits

If you have a partial or full security exit, you can correlate access to UDBs by sharing information between your external security exit and the NCL authorization exit.

The NEXCORR field of the communications area in the NCL authorization exit is the standard correlator. Your external security exit can be use this correlator. This correlator could, for example, be used to address a control block that contains information associated with that user.

If your external security exit supports a SAF user token, then this token is also available in the NEXUTOKN field in the communications area. This token is the UTOKEN provided by the external security exit when the user logs on. The token can be used in a security authorization call to verify the user's access to the UDB.

### Sample Distributed NCL Authorization Exit

A sample exit, NCLEX01, is supplied in the SMP target zone library, *dsnpref*.*pvpref*.CC2DSAMP.

***dsnpref***

Is your site-specific data set name prefix.

***pvpref***

Is your product version prefix.

# INMC Link Security

A security exit is provided as one of the facilities of INMC (Inter-Network Management Connection). This exit provides security for the connections between domains.

The security exit is coded as an assembler language exit routine. An exit must exist in each of the domains that are being linked. The exits are installed with the following components:

Primary exit

> Communicates with the secondary exit in the other domain. The exit decides whether to establish the link and allow traffic flow.

Secondary exit

> Responds to messages received from the primary exit. The exit makes no decisions whether a link is established or terminated.

This design allows each end of an INMC link to determine independently whether to activate a link, without having to depend on cooperation from the remote domain to enforce the decision.

These components are described in more detail in the following sections.

## Primary Exit

The INMC *primary* exit decides whether a newly opened INMC link to a remote domain should be made available for general traffic. The secondary exit is not involved in this decision.

In order to pass control to the exit to make this decision the following calls are made by INMC to the exit:

- **The Initialization Call**—made when an INMC link to another domain is activated. The primary exit is notified of the event and whether the remote domain is configured with a secondary exit.
- **The Deliver Call**—made to the primary exit when INMC receives the reply from the (remote) secondary exit to a message sent to it earlier by the primary exit.
- **The Notify Call**—made to the primary exit when INMC determines that an unexpected event (for example, link outage) has occurred, which results in the deactivation of the link.

For every piece of data sent by the primary exit, there is always a response from the secondary exit. This response can be one of the following:

- Data exchange is complete.
- Link outage has occurred and no reply is possible.

If any protocol errors occur, a hang or stalemate condition might occur between the primary and secondary exits. Care, therefore, must be taken when designing the flow of information between the exit pairs and the rules defined for the various parameter lists passed to the exits must be adhered to.

**More information:**

INMC Security Exit Support (see page 199)

## Secondary Exit

The function of the *secondary* exit is to act solely as a respondent to any messages received from the primary exit in the remote domain. The secondary exit has no power to recommend activation or closure of the link.

Calls made to the secondary exit are as follows:

- **The Initialization Call**—made when an INMC link becomes active. The exit can then perform any initialization required before returning to INMC.
- **The Deliver Call**—made to the secondary exit to deliver a message from the primary exit. The secondary exit must respond with a reply to the message, even if that reply is a null message.
- **The Disconnect Call**—made if the primary exit decides that the conversation is to be ended or when a link outage occurs.

This call allows the secondary exit to end cleanly and tidy up any allocated resources.

**More information:**

INMC Security Exit Support (see page 199)

# Data Set Allocation Authority

The data set access authorization exit (NMDSNCHK) allows you to check whether a user attempting to allocate a data set dynamically is permitted to do so. This exit is invoked in two situations:

- When an ALLOCATE command is issued
- During FTS transmission processing

The data set access authorization exit is specified in the NMSECURITY parameter group (enter **/PARMS**). The default value is usually NMDSNCHK. If you have implemented the NMSAF solution, the default is NMSECDSN.

You can write your own data set access authorization exit to perform any required security checking, and implement it by updating the NMSECURITY parameter group.

**More information:**

Install the Data Set Access Authorization Exit (see page 190)

## Sample Distributed Exit

A sample data set access authorization exit called NMDSNCHK is supplied in the distribution libraries. It comes in both load module format and source form. The distributed version authorizes all requests. The source contains extensive documentation and can be used as the base product for tailoring to your installation's requirements.

A second exit (called NMSECDSN (see page 26)) is supplied for use with NMSAF.

**More information:**

Data Set Authorization Exits Support (see page 187)

# Security for Data Set Services

The data set services authorization exit, NMDSSCHK, is called periodically during data set services processing to perform security-related functions. You can use this exit to implement security procedures at the user level rather than at the region level.

The data set services authorization exit is similar to the data set access authorization exit. The exit is used by data set services functions, including the data set services ALLOC and UNALLOC operations.

The data set access authorization exit is specified in the NMSECURITY parameter group (enter **/PARMS**). The default value is usually NMDSSCHK. If you have implemented the NMSAF solution, the default is NMSECDSS.

You can write your own data set access authorization exit to perform any required security checking, and implement it by updating the NMSECURITY parameter group.

**Note:** For information about the data set services interface ($DSCALL) see the *Network Control Language Reference*.

**More information:**

Install the Data Set Services Authorization Exit (see page 197)

## Sample Distributed Exit

A sample data set services authorization exit called NMDSSCHK is supplied in the distributed libraries. This is in load module format and in source form. This distributed version authorizes all requests and contains extensive documentation. The sample exit is provided as a base for you to implement your own exit procedure.

A second exit (called NMSECDSS (see page 26)) is supplied for use with NMSAF.

**More information:**

Data Set Authorization Exits Support (see page 187)

# Appendix A: SXCTL Parameters

The SXCTL parameter file specifies the parameters that you use for the NMSAF security solution (see page 19).

This section contains the following topics:

SXCTL Parameters (see page 67)

## SXCTL Parameters

The NMSAF security exit reads the SXCTL file during initialization of your region:

- Blank lines and lines with an asterisk (*) as the first non-blank character are ignored.
- Other lines must contain a valid SXCTL parameter.

You can specify any of these parameters in the SXCTL file.

**APPCCHECK { NO | <u>YES</u> }**

Controls whether APPC user sessions are validated against security.

**Note:** Do not set this parameter to NO.

**APPCMODEL { <u>NO</u> | YES }**

Controls whether an APPC user is eligible for model processing (if not known by this region).

**NO**

(Default) The logon is rejected.

**YES**

A model can be used (subject to model processing rules).

**CHANGEPWD { NO | YES }**

Controls the use of the Password Change facility in your region.

**NO**

(Default) Blocks attempts to use the UAMS Password Change facility, or any other password change interface (for example, using EASINET), and produces an error message. This setting prevents users from using these region features to change their passwords (whether in UAMS or external security). It can be useful in distributed security environments where passwords must be changed by using a particular mechanism.

**YES**

Allows the Password Change facility to be used (although the security system can reject or ignore it).

**CONCHECK { YES | NO }**

Controls the checking of console user IDs. These are user IDs for system consoles.

**YES**

(Default) The console user ID is presented to SAF.

**NO**

The console user ID is not presented to SAF.

**Note:** If CONCHECK YES is specified, this user ID is presented before the CONUID user ID is presented, if one is set.

**CONUID { - | *userid* }**

Provides a single SAF user ID to use for all console environments for this region. This parameter can prevent the need to define individual console users to the security system. If CONCHECK YES is set, the value of CONUID is presented to SAF only if verification of the specific console user ID failed.

**-**

Clears the value (blank).

***userid***

Specifies the user ID.

**Limits:** One through eight characters, with all characters alphanumeric or national

**Note:** Regardless of the settings of CONCHECK and CONUID, the logon procedure ignores a failure of a console user logon. The procedure allows the logon and, if the user is also not defined on UAMS, supplies default values.

**DSSDSSEC { NO | YES }**

Controls whether data set services register system users for data set resource checking. This feature requires the NMSECDSS exit to be active.

**DSSDUSEC { YES | NO }**

Controls whether data set services register normal users for data set resource checking. This feature requires the NMSECDSS exit to be active.

**DSSHSSEC { NO | YES }**

Controls whether data set services register system users for HFS file resource checking. This feature requires the NMSECDSS exit to be active.

**DSSHUSEC { YES | NO }**

Controls whether data set services register normal users for HFS file resource checking. This feature requires the NMSECDSS exit to be active.

**MODEL { NO | SYSPARM | SINGLE | LIST }**

Controls the use of the MODEL user facility.

**NO**

(Default) Specifies that no modeling is performed.

**SYSPARM**

Specifies that the setting of SYSPARMS MODLUSER be used.

**SINGLE**

Specifies that if a model name is specified in SXCTL, it is used as the model.

**LIST**

Specifies that if a resource or model list is defined, then it is used to determine the model name.

Modeling applies only if a user logs on to the region and no UAMS definition exists. You can control which logon types can participate in modeling.

**MODELGROUP { *saf.resource.name* | * } *modelname***

Supplies an entry in a list of SAF resource names and associated model names. The parameter can be repeated up to 20 times in the SXCTL file. The order in which the pairs of resource names and model names are specified is the order in which the resource names are tested. Specifying a resource name of * always matches (no SAF AUTH call is made).

If MODEL LIST is specified and modeling is required (that is, the user is not known to the region), then each defined resource name is tested (using the class as set by the RCLASS parameter) in turn, until a resource is found that the user has at least READ access to (or the * entry is reached). If a match is found, the associated model name is returned. If no match is found (and no * entry is found), then no model name is returned and the logon is rejected.

***saf.resource.name modelname***

Must be in valid PDSNAME format. The length must be one through eight characters; the first character must be alphabetic or national (@,#,$) and the rest must be alphanumeric or national.

**MODELNAME { - | *userid* }**

Supplies the model name to use for modeling if MODEL SINGLE is specified (otherwise it is ignored). If no model name is specified (the default), it is as if MODEL NO is specified.

**-**

Clears the value (blank). This setting can cause substitution by a default value.

***userid***

Names the model.

**Limits:** One through eight characters, with all characters alphanumeric or national

**RAPPL { - | *name* }**

Sets the APPL value to use on RACROUTE calls.

**-**

(Default) A dash means none; the primary ACBNAME is then used.

***name***

Must be in valid PDSNAME format. The length must be one through eight characters; the first character must be alphabetic or national (@,#,$) and the rest must be alphanumeric or national.

**RCLASS { - | *name* }**

Sets the SAF resource class to use for most RACROUTE AUTH checks (for example, for model determination).

**-**

(Default) A dash (-) means none; FACILITY is then used.

***name***

Must be in valid PDSNAME format. The length must be one through eight characters; the first character must be alphabetic or national (@,#,$) and the rest must be alphanumeric or national.

**ROFCHECK { YES | NO }**

Controls the SAF validation of a ROF (Remote Operator Facility) user. ROF users are users that use the SIGNON and ROUTE commands from a remotely connected region to send commands to this one. The user ID is always the user ID that the user originally signed on with.

**YES**

(Default) Validates the user by a SAF call. If the user is not known (or has been revoked, for example), the signon fails.

**NO**

Makes no SAF call on this system for a ROF user.

**ROFMODEL { NO | YES }**

Controls whether a ROF user is eligible for model processing (if not known by this region).

**NO**

(Default) The logon is rejected.

**YES**

A model can be used (subject to model processing rules).

**ROFPWD { YES | NO }**

Controls whether a password is required when signing on to this region by using the ROF SIGNON command.

**YES**

(Default) The correct SAF password (for this region’s security system) for the current user ID must be supplied on the SIGNON command; otherwise, the signon is rejected.

**NO**

No password is required (SAF is asked to validate the user with no password if none is supplied).

**Note:** Specifying ROFPWD YES can cause problems with system user IDs. If NCL processes executing in these environments issue ROF signons to other systems, then, when the requests come in, the user ID is not treated as a system user. Normal validation occurs. This scenario can be a problem if a password is required.

**SYSCHECK { YES | NO }**

Controls the checking of system (or background) user IDs; for example, the BSYS and BLOG users, and the PPOP and AOMP regions.

**Note:** If SYSCHECK YES is specified, this user ID is presented before the SYSUID user ID is presented, if one is set.

**YES**

(Default) The user ID is presented to SAF for validation (no password is required). If SAF verifies the user ID, then it is accepted.

**NO**

The generated user ID is not presented to SAF.

**SYSUID { - | *userid* }**

This parameter provides a single SAF user ID to use for all the system (or background) user IDs for this region. This feature prevents the need to define multiple user IDs (such as NM01BSYS and NM01BMON) to the security system. If SYSCHECK YES is set, the value of SYSUID is presented to SAF only if verification of the specific system user ID failed.

**-**

Clears the value (blank). This setting can cause substitution by a default value.

***userid***

Specifies the user ID.

**Limits:** One through eight characters, with all characters alphanumeric or national

**Note:** Regardless of the settings of SYSCHECK and SYSUID, the initialization procedure ignores a failure of a system user logon. The procedure continues initializing and, if the user is also not defined on UAMS, supplies default values.

**TRACE { NO | YES }**

Enables tracing to the SXTRACE data set.

**NO**

Disables all tracing, regardless of other trace options.

**YES**

Enables tracing (provided the SXTRACE file can be opened during initialization), but other trace options must be set to cause actual tracing to occur.

**TRACEMOD { NO | YES }**

Enables tracing of the security exit module flow. Typically, this feature is used only if requested by Technical Support to track down errors in the exit.

**Note:** This option produces a large amount of trace output.

**TRACEPL { NO | YES }**

Enables tracing of the security exit call parameter list on entry and exit. The trace includes the fields pointed to by parameters that are not null (except passwords).

**TRACESAF { NO | ERROR | YES }**

Enables tracing of the results of RACROUTE (SAF) macro calls.

**NO**

Disables all tracing.

**ERROR**

Causes tracing of those RACROUTE calls that failed in some way.

**YES**

Traces all RACROUTE calls. The trace includes the parameter list and return codes.

**TSOMODEL { NO | YES }**

Controls whether a TSO user is eligible for model processing (if not known by this region).

**NO**

(Default) Specifies that automatic model processing is not used and the user (if not defined to UAMS) is presented with a blank logon panel that uses normal logon processing rules.

**YES**

Means that a model can be used (subject to model processing rules).

**TSOPWD { <u>YES</u> | NO }**

Controls the requirement for a password when using the TSO pass through facility (the NMLOGON TSO command).

**YES**

(Default) Specifies that the user is presented with a normal logon screen and must enter the user ID and password to gain access.

**NO**

Specifies that the user can be logged on (using the current TSO user ID) with no password (if this logon is not blocked in the UAMS definition for this user).

**USERFLAG*n* { <u>NO</u> | YES }**

These parameters (up to 8) set flags in the global area accessible to other exits. They can be used to control logic in installation-written exits, such as NCLEX01.

**USERNAME*n* { <u>-</u> | *name* }**

These parameters (up to 4) set name values in the global area accessible to other exits. They can be used as input data in installation-written exits, such as NCLEX01.

**-**

Clears the value (blank). This setting can cause substitution by a default value.

***name***

Must be in valid PDSNAME format. The length must be one through eight characters; the first character must be alphabetic or national (@,#,$) and the rest must be alphanumeric or national.

**USERUID*n* { - | *uid* }**

These parameters (up to 4) set user ID values in the global area accessible to other exits. They can be used as input data in installation-written exits (such as NCLEX01).

**-**

Clears the value (blank). This setting can cause substitution by a default value.

***uid***

Specifies a user ID.

**Limits:** One through eight characters, with all characters alphanumeric or national

**VAPPCLINK { <u>NO</u> | YES | N12 | N21 | BOTH }**

Controls the activation of the APPC link security facility. The facility uses a SAF query to extract a password, with a resource class of APPCLU.

**NO**

Disables the facility. No passwords are returned.

**YES**

Causes a SAF resource query using network.locallu.remotelu to be performed. If the query works, the password is returned.

**N12**

The same as YES.

**N21**

Causes a SAF resource query using network.remotelu.locallu to be performed. If the query works, the password is returned.

**BOTH**

Causes a SAF resource query using network.locallu.remotelu to be performed, and then another SAF resource query using network.remotelu.locallu. If either of these queries works, the password is returned.

**Note:** Advanced Program-to-Program Communication (APPC) supports the use of link-level passwords. Both the DEFLINK and LINK START commands for APPC allow the specification of a password, or alternatively the use of PASSWORD=EXIT, which means that the security exit can return the password.

**WEBMODEL { NO | YES }**

Specifies whether a WebCenter user not known by this region is eligible for model processing.

**NO**

(Default) The logon is rejected.

**YES**

A model can be used (subject to model processing rules).

**More information:**

How You Implement NMSAF (see page 21)
Security for File Access (see page 58)

# Appendix B: Structured Fields

This section contains the following topics:

## Overview

Each structured field identifies an item of information that can be specified when a user ID is defined. For example, user name, user location, and so on. These structured fields provide the same information as the fields in the UAMS function for defining a user ID.

Structured fields are used to exchange information between security exits and system services. When an exit returns user ID information to system services it is always in the form of a list of pointers to one or more structured fields. By processing the fields returned by the exit, system services build an internal representation of the user ID.

Your system services also translate this internal representation of a user ID into the same set of structured fields when a request is passed to an exit.

**More information:**

Working with UAMS (see page 27)

# Format of Structured Fields

The format of a structured field is as follows:

**Bytes 00 through 01**

Specifies the field key—a hexadecimal value that defines the item of information described by this field.

**Bytes 02 through 03**

Specifies the subfield count—a hexadecimal count of the number of subfields within this structured field.

**Bytes 04 through 05**

Specifies the subfield length—the hexadecimal length of the following subfield.

**Bytes 06 through *nn***

Specifies the subfield data.

Subfield length and subfield data are repeated according to the number specified in subfield count.

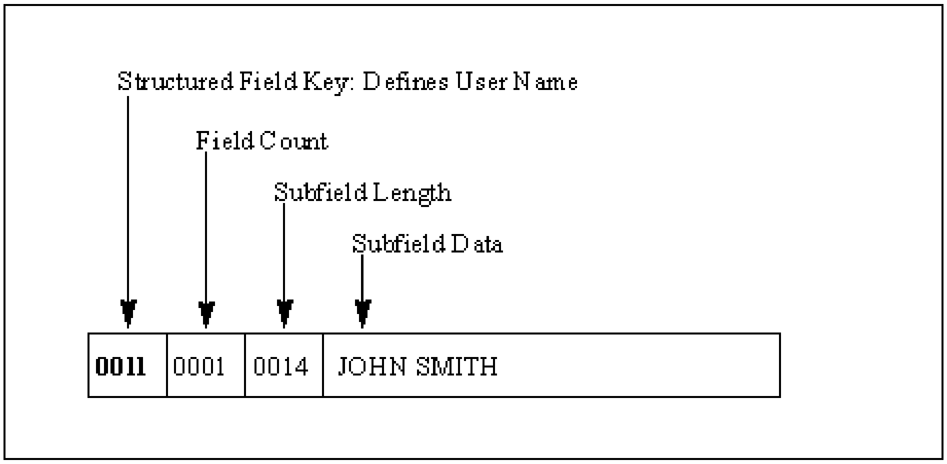
**Note:** For examples of how to update structured fields, including those fields with multiple subfields, see the description of the &SECCALL verb in the *Network Control Language Reference*.

### Example:

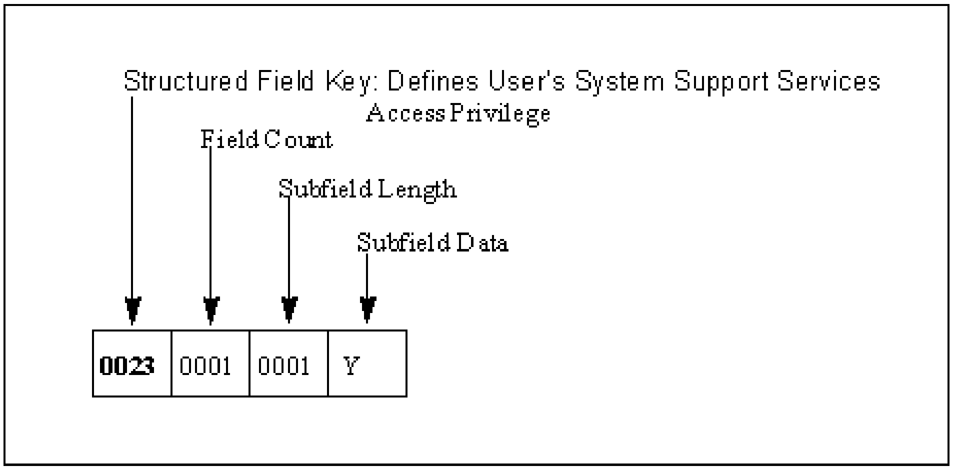
The following diagram shows structured field X’0011’. This field is used to define a user’s name. This name can be associated with the user ID so that the person can be identified when they log on.

Structured field X’0011’ consists of:

- Field key of X‘0011’
- Subfield count of X‘0001’ (1 subfield)
- Subfield length of X‘0014’ (20 characters)
- Subfield data of JOHN SMITH (padded with blanks on the right to make up the 20 characters)



The following diagram shows structured field X‘0023’. This field too has a single subfield with a value of Y and is used to indicate that the user is entitled to access the System Support Services function.



## Structured Field Error Conditions

Error conditions can occur during the processing of a set of structured fields due to one of the following reasons:

- The omission of a structured field.
- An invalid value has been specified in a structured field.

If an error does occur during structured field processing, then:

- The structured field is assigned its default value.
- An error message is written to the log.

If error conditions occur, a user might not have all the expected privileges when logged on to the system.

## Structured Field Sequences

Your product region processes structured fields in the order in which they appear. While most structured fields are independent of other fields, certain fields must be processed before others. Where such dependencies exist, they are noted in the description of the fields concerned.

# Structured Field Descriptions

The description of each field includes comments on special processing or validation requirements.

**Note:** The length of each subfield is fixed and cannot be varied. Where applicable, data fields must be blank padded to the full length of the subfield.

## SF X‘0010’—User ID Name

| | |
|---|---|
| FUNCTION: | Defines user ID name |
| KEY: | X‘0010’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

The user ID name is any 8 character string forming a valid user ID. The name must be left justified and blank-padded to a length of 8 characters. This field is optional. If generated by the exit in response to a call for user ID information, it must be the same user ID name as that requested by your product region. If it is omitted, it defaults to the user ID name requested.

## SF X‘0011’—User Name

| | |
|---|---|
| FUNCTION: | Defines user name |
| KEY: | X‘0011’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0014’ |
| SUBFIELD 1 DATA: | CL20‘ ’ |

The user name is any 20 character string. It allows the name of the user (or any other comments) to be displayed at log on time. This field is optional. If omitted, the name field defaults to blanks. If provided, the subfield must be 20 characters long.

## SF X‘0012’—User Location

| | |
|---|---|
| FUNCTION: | Defines user location |
| KEY: | X‘0012’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0014’ |
| SUBFIELD 1 DATA: | CL20‘ ’ |

The user location is any 20 character string. This field is optional. If omitted, the location field defaults to blanks. If provided, the subfield must be 20 characters long.

## SF X‘0013’—User Telephone Number

| | |
|---|---|
| FUNCTION: | Defines user telephone number |
| KEY: | X‘0013’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0014’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

The user telephone number field is any 32 character string. This field is optional. If omitted, the field will default to blanks. If provided, the subfield must be 32 characters long.

## SF X‘0014’—User Language Code

| | |
|---|---|
| FUNCTION: | Defines user language code |
| KEY: | X‘0014’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0002’ |
| SUBFIELD 1 DATA: | CL2‘ ’ |

The user language code field is any 2 character string. This field is optional. If omitted, the field will default to blanks. If provided, the subfield must be 2 characters long. If omitted, the field defaults to the current setting of the system language code set by the SYSPARMS LANG operand.

## SF X‘0015’—User ID Suspend Date

| | |
|---|---|
| FUNCTION: | Defines user ID suspend date |
| KEY: | X‘0015’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0006’ |
| SUBFIELD 1 DATA: | CL6‘ ’ |

This field defines the date on which the user ID is to be suspended from further ability to log on to the system. Format is YY.DDD. If omitted, the field defaults to blanks.

## SF X‘0016’—Terminal Restrictions

| | |
|---|---|
| FUNCTION: | Defines terminal restrictions |
| KEY: | X‘0016’ |
| SUBFIELD COUNT: | X‘0003’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0008’ |
| SUBFIELD 2 DATA: | CL8‘ ’ |
| SUBFIELD 3 LENGTH: | X‘0008’ |
| SUBFIELD 3 DATA: | CL8‘ ’ |

Subfields 1, 2 and 3 define the terminals to which this user ID is restricted, if any. Fields can be left blank, or the field omitted if terminal restrictions do not apply to the user ID.

**Note:** The full security exit is responsible for performing the actual terminal restriction processing. Your product region performs no terminal restriction validation when a full security exit is utilized.

## SF X‘0017’—Time-out Control

| | |
|---|---|
| FUNCTION: | Defines time-out control |
| KEY: | X‘0017’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user ID is subject to time-out facilities. Valid values are Y or N. If the field is omitted, the default is Y.

## SF X‘0018’—Date/Time User ID Last Updated

| | |
|---|---|
| FUNCTION: | Defines date/time user ID last updated |
| KEY: | X‘0018’ |
| SUBFIELD COUNT: | X‘0002’ |
| SUBFIELD 1 LENGTH: | X‘0014’ |
| SUBFIELD 1 DATA: | CL20‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0010’ |
| SUBFIELD 2 DATA: | CL16‘ ’ |
| SUBFIELD 3 LENGTH: | X‘0008’ |
| SUBFIELD 3 DATA: | CL8‘ ’ |

**Subfield 1**

- **Value**—Date and time that this user ID last logged on to system
- **Format**—15-character last logged on date, for example, MON 09-FEB-2006, and a 5-character last logged on time, for example, 08.30

**Subfield 2**

- **Value**—Date and time that this user ID was last updated
- **Format**—11-character last updated date, for example, 09-FEB-2006, and a 5-character last logged on time, for example, 08.30

**Subfield 3**

- **Value**—User ID of the user who last updated this field
- **Format**—Eight characters

## SF X‘0019’—Multiple Signon Capability

| | |
|---|---|
| FUNCTION: | Defines multiple signon capability |
| KEY: | X‘0019’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field determines whether the user ID can log on multiple times from a terminal or MAI environment. Valid values are Y or N only. If omitted, the default is N.

## SF X‘001A’—Group Definition for User

| | |
|---|---|
| FUNCTION: | Defines group definition to be used for this user |
| KEY: | X‘001A’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

This field defines the group definition that is used by this user ID. This definition is read for all functions related to the user ID. The privileges of the group definition are applied to this user ID. Only user information (that is, name, location, phone number and initial OCS command) is specified as unique to this user ID. Valid values are:

- Blank or nulls—indicates that there is no group definition for this user
- Group name—indicates the group definition that is to be used for this user ID

## SF X‘001B’—User ID Definition Type

| | |
|---|---|
| FUNCTION: | User ID definition type |
| KEY: | X‘001B’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field indicates whether this is a Group definition or a User definition. Valid values are Y (group definition) or N (user definition). If omitted, the default is N (user definition).

## SF X‘001C’—User Password Expiry Action Indicator

| | |
|---|---|
| FUNCTION: | User password expiry action indicator |
| KEY: | X‘001C’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field indicates whether password expiry checking is to be performed for this user ID. Valid values are Y (password renewal at installation determined expiry interval is enforced) or N (no expiry checking is performed). If omitted, the default is Y.

## SF X‘001D’—User Email Address

| | |
|---|---|
| FUNCTION: | Defines user email address |
| KEY: | X‘001D’ |
| SUBFIELD COUNT: | X‘0002’ |
| SUBFIELD 1 LENGTH: | X‘0040’ |
| SUBFIELD 1 DATA: | CL64‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0080’ |
| SUBFIELD 2 DATA: | CL128‘ ’ |

**Subfield 1**

The user email address field is any 64-character string. This field is optional. If omitted, the default is blanks. If provided, the subfield must be 64 characters long. The field may contain asterisks as placeholders for User ID and domain name.

**Subfield 2**

The expanded user email address field is a copy of the user email address, with any asterisk placeholders replaced as follows:

- An asterisk before '@' is replaced by the UAMS User ID.
- An asterisk after '@' is replaced by the contents of the system parameter EMAILDMN as set in the SYSTEMID Customizer Parameter Group.

This field is not updateable.

## SF X‘001E’—Model User ID Name

| | |
|---|---|
| FUNCTION: | Defines model user ID |
| KEY: | X‘001E’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

The Model User ID name is any eight-character string forming a valid user ID. The name must be left-justified and blank-padded to a length of eight characters. This field is optional. It is set if the UAMS record is generated from a model record.

## SF X‘0020’—OCS Access Privilege

| | |
|---|---|
| FUNCTION: | Defines OCS access privilege |
| KEY: | X‘0020’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field indicates whether the user has access to OCS. Valid values are Y or N. If omitted, or an invalid value is specified, the default is N.

## SF X‘0021’—Broadcast Services Privilege

| | |
|---|---|
| FUNCTION: | Defines Broadcast Services privilege |
| KEY: | X‘0021’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field indicates whether the user has access to Broadcast Services. Valid values are Y or N. If omitted, or an invalid value is specified, the default is N.

## SF X‘0022’—Network Services Access Privilege

| | |
|---|---|
| FUNCTION: | Defines network services access privilege |
| KEY: | X‘0022’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user ID has access to the Network Management options. Valid values are Y or N. If omitted, or an invalid value is specified, the default is N. This privilege is a prerequisite for other privileges which are available to the user within Network Services, for example, TCP/IP Services privilege.

This structured field is a prerequisite for the following structured fields:

- SF0026—NEWS Access Privilege
- SF002D—NCS Access Privilege
- SF0090—NCPView Access Privilege
- SF0150—NEWS Reset Privilege
- SF0151—NTS Access Privilege
- SF0530—TCP/IP Services Access Privilege

## SF X‘0023’—System Support Privilege

| | |
|---|---|
| FUNCTION: | Defines System Support privilege |
| KEY: | X‘0023’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field indicates whether the user has access to System Support Services. Valid values are Y or N. If omitted, or an invalid value is specified, the default is N.

## SF X'0025'—CA SOLVE:FTS Access Privilege

| | |
|---|---|
| FUNCTION: | Defines CA SOLVE:FTS access privilege |
| KEY: | X'0025' |
| SUBFIELD COUNT: | X'0001' |
| SUBFIELD 1 LENGTH: | X'0001' |
| SUBFIELD 1 DATA: | CL1' ' |

This field indicates whether the user is entitled to access CA SOLVE:FTS. Valid values for the subfield data are Y or N. N is assumed if the field is omitted or specifies a value other than Y. This field is ignored if the system is not configured with the CA SOLVE:FTS product.

## SF X‘0026’—NEWS Access Privilege

| | |
|---|---|
| FUNCTION: | Defines CA NetMaster Network Management for SNA NEWS access privilege |
| KEY: | X‘0026’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field indicates whether the user has access to NEWS. Valid values are Y or N. The default is N.

This field is ignored if the user ID has not been given access privileges to Network Management, that is, if the user ID does not have structured field key X‘0022’.

## SF X‘0027’—MAI-FS Access Privilege

| | |
|---|---|
| FUNCTION: | Defines CA SOLVE:Access Session Management MAI-FS access privilege |
| KEY: | X‘0027’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field indicates whether the user has access to MAI-FS. Valid values are Y or N. If omitted, or an invalid value is specified, the default is N. Specifying Y in this field allows the user to issue SETMODE MAI and use the MAISESS command. In addition, the supplied primary menu offers an option to select the MAI-FS primary menu.

## SF X‘0028’—User Services Procedure Name

| | |
|---|---|
| FUNCTION: | Defines User Services procedure name |
| KEY: | X‘0028’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

This field identifies the 8-character name of the NCL procedure to be invoked when the user selects the User Services option from the Primary Menu. The name specified must be a valid NCL procedure name, padded with blanks to 8 characters if required. Imbedded blanks within the name cause truncation.

This field is only valid if the user has access privileges for User Services. If this field is omitted, no User Services procedure name is assigned to the user ID.

## SF X‘0029’—User’s NCL Procedure Library

| | |
|---|---|
| FUNCTION: | Defines user’s NCL procedure library |
| KEY: | X‘0029’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

This field provides the DD name in the JCL (z/OS) or the filetype (z/VM) that defines the NCL procedure library to be used by this user. Whenever the user executes an NCL procedure, this library is searched for the procedure. If omitted, the library defined by the COMMANDS DD statement is used.

## SF X‘002A’—UAMS Access Privilege

| | |
|---|---|
| FUNCTION: | Defines UAMS access privilege |
| KEY: | X‘002A’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user ID has access to the UAMS maintenance functions. Valid values are Y or N. If omitted, or if an invalid value is specified, the default is N.

## SF X‘002B’—Operations Management Privilege

| | |
|---|---|
| FUNCTION: | Defines Operations Management privilege |
| KEY: | X‘002B’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user ID has access to the facilities of Operations Management. Valid values are Y or N. The default is N.

## SF X‘002C’—TSO Autologon Privilege

| | |
|---|---|
| FUNCTION: | Defines TSO auto-logon privilege |
| KEY: | X‘002C’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user ID can automatically log on to the system using an EIP component, such as a TSO address space, without having to reenter their user ID/password combination. If omitted, the default is N. This field is only valid if you are licensed for EIP.

## SF X‘002D’—NCS Access Privilege

| | |
|---|---|
| FUNCTION: | Defines NCS access privilege |
| KEY: | X‘002D’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user ID has access to the facilities of NCS. Valid values are Y or N. If omitted, or an invalid value is specified, the default is N.

This field is ignored if the user ID has not been given access privileges to Network Management, that is, if the user ID does not have structured field key X‘0022’.

## SF X‘002E’—User’s SPLIT/SWAP Privilege

| | |
|---|---|
| FUNCTION: | Defines user’s SPLIT/SWAP privileges |
| KEY: | X‘002E’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user ID is entitled to use the SPLIT/SWAP facility for operating two screen windows. Valid values are Y or N. If the field is omitted, the default is Y.

## SF X‘002F’—Library Services Path Name

| | |
|---|---|
| FUNCTION: | Defines Library Services path name |
| KEY: | X‘002F’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

This field defines the user’s library path name. If a library path name is to be used to control the sequence of access to the panels data sets, enter the name of the path name to be used. See the LIBRARY command for information about the path name definition. The default value is blanks, indicating that no path name is to be used.

## SF X‘0030’—User’s Time Zone Name

| | |
|---|---|
| FUNCTION: | Defines user’s time zone name |
| KEY: | X‘0030’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

This field defines the time zone name for the user. If omitted, the system time zone name is used when the user logs on.

## SF X‘0050’—OCS Command Authority Level

| | |
|---|---|
| FUNCTION: | Defines OCS command authority level |
| KEY: | X‘0050’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0003’ |
| SUBFIELD 1 DATA: | CL3‘ ’ |

This field defines the command authority level assigned to the user when operating from OCS or when issuing commands within an NCL procedure. Valid values are 0 to 255 inclusive. If omitted, or an invalid value is specified, the default is 0.

## SF X‘0051’—OCS Monitor Status

| | |
|---|---|
| FUNCTION: | Defines OCS Monitor status |
| KEY: | X‘0051’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user is assigned Monitor status when operating in OCS. Valid values are Y or N. If omitted, or an invalid value is specified, the default is N.

## SF X‘0052’—NPF Command Member

| | |
|---|---|
| FUNCTION: | Defines NPF command member |
| KEY: | X‘0052’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

This field defines the name of the NPF control member that specifies command partitioning for the user ID. The NPF member name supplied must be padded to 8 characters, if necessary.

## SF X‘0053’—MSGPROC Member

| | |
|---|---|
| FUNCTION: | Defines MSGPROC member |
| KEY: | X‘0053’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

This field defines the name of the NCL procedure to be invoked for each OCS window operated by the user. This field only applies if the user requires a MSGPROC NCL procedure to intercept all messages delivered when operating in OCS. The MSGPROC name supplied must be padded to 8 characters, if necessary. No MSGPROC procedure is invoked for the user if this field is omitted.

## SF X‘0054’—OCS Mode Initial Command

| | |
|---|---|
| FUNCTION: | Defines OCS mode initial command |
| KEY: | X‘0054’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0028’ |
| SUBFIELD 1 DATA: | CL40‘ ’ |

This field defines the command that is executed automatically for this user on entry to OCS. If specified, the field data must be 40 characters long, padded with blanks if required. No validation of the command string is performed. If this field is omitted, no initial command is assigned to the user ID.

## SF X‘0055’—PPO Message Receipt Option

| | |
|---|---|
| FUNCTION: | Defines PPO message receipt option |
| KEY: | X‘0055’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field determines whether the user is entitled to receive unsolicited VTAM PPO messages, when logged on to OCS. Valid values are Y or N. The default is N.

## SF X‘0056’—PPO Severity Level

| | |
|---|---|
| FUNCTION: | Defines PPO severity level |
| KEY: | X‘0056’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0004’ |
| SUBFIELD 1 DATA: | CL4‘ ’ |

This field indicates the severity level of PPO messages to be delivered to this user. This field only applies if the user is to have unsolicited VTAM PPO messages delivered when operating in OCS. Valid values are INFO, NORM, WARN, or SER. If omitted, or an invalid value is specified, the default is NORM.

## SF X‘0057’—NPF Message Restriction Option

| | |
|---|---|
| FUNCTION: | Defines NPF message restriction option |
| KEY: | X‘0057’ |
| SUBFIELD COUNT: | X‘0002’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0001’ |
| SUBFIELD 2 DATA: | CL1‘ ’ |

This field defines the NPF message member that specifies the nodes from which the user receives unsolicited VTAM PPO messages.

This field has 2 subfields. The first subfield specifies the name of the NPF member. The second subfield determines whether the user is to receive undeliverable messages: that is, unsolicited messages that refer to network resources which are outside the user’s normal NPF restrictions but which cannot be delivered to any other user. Valid values are Y or N. The default is N.

This field is ignored if the user ID has not been authorized for PPO message receipt, that is, if the user ID does not have structured field key X‘0055’.

## SF X‘0058’—Message Code Value

| | |
|---|---|
| FUNCTION: | Defines message code value |
| KEY: | X‘0058’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | XL1‘ ’ |

This field defines the one-byte bit string representing the message level mask for this user if they have a message code set. Any hexadecimal value (00 to FF) is Valid. If omitted, this field defaults to X‘00’.

## SF X‘0059’—OCS MSG Message Receipt

| | |
|---|---|
| FUNCTION: | Defines OCS MSG message receipt |
| KEY: | X‘0059’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field specifies the default PROFILE MSG command when entering OCS. The default is N.

## SF X‘005A’—OCS Unsolicited Message Receipt Option

| | |
|---|---|
| FUNCTION: | Defines OCS unsolicited message receipt option |
| KEY: | X‘005A’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field specifies the default PROFILE UNSOL command when entering OCS. The default is N.

## SF X‘005B’—Resource List Member

| | |
|---|---|
| FUNCTION: | Defines resource list member |
| KEY: | X‘005B’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

This field defines the resource list member that specifies the combined NPF restrictions for resource display and affect through VTAM commands. If omitted, the field defaults to blanks.

## SF X‘005C’—User Time-out (1) Period

| | |
|---|---|
| FUNCTION: | Defines user time-out (1) period |
| KEY: | X‘005C’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0006’ |
| SUBFIELD 1 DATA: | CL6‘ ’ |

This field defines the interval used as the first time-out period for an inactive user ID. The format of the interval can be *mmm.ss* (minutes and seconds) or *mmm* (minutes). This field overrides SYSPARMS TOTIME1.

## SF X‘005D’—User Time-out (2) Period

| | |
|---|---|
| FUNCTION: | Defines user time-out (2) period |
| KEY: | X‘005D’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0006’ |
| SUBFIELD 1 DATA: | CL6‘ ’ |

This field defines the interval used as the second time-out period for an inactive user ID. The format of the interval can be *mmm.ss* (minutes and seconds) or *mmm* (minutes). This field overrides SYSPARMS TOTIME2.

## SF X‘005E’—User Time-out (1) Action

| | |
|---|---|
| FUNCTION: | Defines user time-out (1) action |
| KEY: | X‘005E’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0028’ |
| SUBFIELD 1 DATA: | CL40‘ ’ |

This field defines the action to be performed when a user ID’s first time-out interval expires. The action can be any command string, for example, LOCK, DISC, CANCEL LU=*. These commands have equivalents in the SYSPARMS time-out actions. A value of NONE can be used to disable the time-out action. This structured field overrides SYSPARMS TOACT1.

## SF X‘005F’—User Time-out (2) Action

| | |
|---|---|
| FUNCTION: | Defines user time-out (2) action |
| KEY: | X‘005F’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0028’ |
| SUBFIELD 1 DATA: | CL40‘ ’ |

This field defines the action to be performed when a user ID’s second time-out interval expires. The action can be any command string, for example, LOCK, DISC, CANCEL LU=*. These commands have equivalents in the SYSPARMS time-out actions. A value of NONE can be used to disable the time-out action. This structured field overrides SYSPARMS TOACT2.

## SF X‘0060’—User’s APPC Access Key

| | |
|---|---|
| FUNCTION: | Defines user’s APPC access key |
| KEY: | X‘0060’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0002’ |
| SUBFIELD 1 DATA: | BL2‘00’ |

This field defines the user’s APPC access key. The mask contains 16 bits, representing, from right to left, the access codes 1 to 16. These are used, together with the APPC lock to allow this user to act on behalf of another user, without supplying the target user’s password.

A 16-bit key indicating the APPC access key must be specified. If omitted, the field defaults to zeroes.

## SF X‘0061’—User’s APPC Access Lock

| | |
|---|---|
| FUNCTION: | Defines user’s APPC access lock |
| KEY: | X‘0061’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0002’ |
| SUBFIELD 1 DATA: | BL2‘00’ |

This field defines the user’s APPC access lock. The key contains 16 bits, representing, from right to left, the access codes 1 to 16. These are used, together with the APPC key, to allow another user to act on behalf of this user, without supplying this user’s password. If omitted, the field defaults to zeroes. Valid value is a 16-bit mask indicating the APPC access lock.

## SF X‘0070’—Installation Attribute Field 1

| | |
|---|---|
| FUNCTION: | Installation Attribute field 1 |
| KEY: | X‘0070’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 1 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0071’—Installation Attribute Field 2

| | |
|---|---|
| FUNCTION: | Installation Attribute field 2 |
| KEY: | X‘0071’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 2 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0072’—Installation Attribute Field 3

| | |
|---|---|
| FUNCTION: | Installation Attribute field 3 |
| KEY: | X‘0072’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 3 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0073’—Installation Attribute Field 4

| | |
|---|---|
| FUNCTION: | Installation Attribute field 4 |
| KEY: | X‘0073’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 4 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0074’—Installation Attribute Field 5

| | |
|---|---|
| FUNCTION: | Installation Attribute field 5 |
| KEY: | X‘0074’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 5 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0075’—Installation Attribute Field 6

| | |
|---|---|
| FUNCTION: | Installation Attribute field 6 |
| KEY: | X‘0075’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 6 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0076’—Installation Attribute Field 7

| | |
|---|---|
| FUNCTION: | Installation Attribute field 7 |
| KEY: | X‘0076’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 7 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0077’—Installation Attribute Field 8

| | |
|---|---|
| FUNCTION: | Installation Attribute field 8 |
| KEY: | X‘0077’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 8 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0078’—Installation Attribute Field 9

| | |
|---|---|
| FUNCTION: | Installation Attribute field 9 |
| KEY: | X‘0078’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 9 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0079’—Installation Attribute Field 10

| | |
|---|---|
| FUNCTION: | Installation Attribute field 10 |
| KEY: | X‘0079’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0020’ |
| SUBFIELD 1 DATA: | CL32‘ ’ |

This field allows specification of data for Installation Attributes assigned to this user ID. The data can be interrogated through NCL. This field corresponds to Installation Attribute field 10 on the Installation Attributes panel in UAMS. If omitted, the field defaults to blanks.

## SF X‘0080’—Access to CA SOLVE:InfoMaster Maintenance Functions

| | |
|---|---|
| FUNCTION: | Defines user’s access to CA SOLVE:InfoMaster maintenance functions |
| KEY: | X‘0080’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user ID has access to CA SOLVE:InfoMaster maintenance functions. Valid values are Y or N. If omitted, the default is N.

## SF X‘0081’—Access to Information Management

| | |
|---|---|
| FUNCTION: | Defines user’s access to Information Management |
| KEY: | X‘0081’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether the user ID has access to Information Management. Valid values are Y or N. If omitted, the default is Y.

## SF X‘0090’—Access to NCPView

| | |
|---|---|
| FUNCTION: | Defines user’s access to CA NetMaster Network Management for SNA NCPView |
| KEY: | X‘0090’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘’ |

This field defines whether a user ID has access to NCPView. Valid values are 0 (not authorized to access NCPView), and 1 (authorized to browse).

This field is ignored if the user ID has not been given access privileges to Network Management, that is, if the user ID does not have structured field key X‘0022’.

## SF X'0100'—CA SOLVE:FTS Definition Privilege

| | |
|---|---|
| FUNCTION: | Defines CA SOLVE:FTS definition privilege |
| KEY: | X'0100' |
| SUBFIELD COUNT: | X'0001' |
| SUBFIELD 1 LENGTH: | X'0001' |
| SUBFIELD 1 DATA: | CL1' ' |

A user who is privileged for CA SOLVE:FTS access can be authorized to create file transmission definitions. This field defines the user's definition privilege. Valid values for the subfield data are P or S, indicating private or system definition privilege. If this field is omitted or specifies an invalid value, no definition privilege is assigned.

This field is ignored if the user ID has not been assigned CA SOLVE:FTS access by the earlier processing of field X'0025'.

## SF X'0101'—CA SOLVE:FTS Private Request Privilege

| | |
|---|---|
| FUNCTION: | Defines CA SOLVE:FTS private request privilege |
| KEY: | X'0101' |
| SUBFIELD COUNT: | X`0001' |
| SUBFIELD 1 LENGTH: | X'0001' |
| SUBFIELD 1 DATA: | CL1'-' |

A user who is privileged for CA SOLVE:FTS access can be authorized to request execution of private file transmissions. This field defines the user's private request privilege. Valid values for the subfield data are Y or N. If this field is omitted or specifies an invalid value, no private request privilege is assigned.

This field is ignored if the user ID has not been assigned CA SOLVE:FTS access by the earlier processing of field X'0025'.

## SF X'0102'—CA SOLVE:FTS System Request Privilege

| | |
|---|---|
| FUNCTION: | Defines CA SOLVE:FTS system request privilege |
| KEY: | X'0102' |
| SUBFIELD COUNT: | X'0001' |
| SUBFIELD 1 LENGTH: | X'0001' |
| SUBFIELD 1 DATA: | CL1' ' |

A user who is privileged for CA SOLVE:FTS access can be authorized to request execution of system file transmissions. This field defines the user's system request privilege. Valid values for the subfield data are Y or N. If this field is omitted or specifies an invalid value, no system request privilege is assigned.

This field is ignored if the user ID has not been assigned CA SOLVE:FTS access by the earlier processing of field X'0025'.

## SF X'0103'—CA SOLVE:FTS Private Control Privilege

| | |
|---|---|
| FUNCTION: | Defines CA SOLVE:FTS private control privilege |
| KEY: | X'0103' |
| SUBFIELD COUNT: | X'0001' |
| SUBFIELD 1 LENGTH: | X'0001' |
| SUBFIELD 1 DATA: | CL1' ' |

A user who is privileged for CA SOLVE:FTS access can be authorized to supervise (control) execution of private file transmissions. This field defines the user's private control privilege. Valid values for the subfield data are Y or N. If this field is omitted or specifies an invalid value, no private control privilege is assigned.

This field is ignored if the user ID has not been assigned CA SOLVE:FTS access by the earlier processing of field X'0025'.

## SF X'0104'—CA SOLVE:FTS System Control Privilege

| | |
|---|---|
| FUNCTION: | Defines CA SOLVE:FTS system control privilege |
| KEY: | X'0104' |
| SUBFIELD COUNT: | X'0001' |
| SUBFIELD 1 LENGTH: | X'0001' |
| SUBFIELD 1 DATA: | CL1' ' |

A user who is privileged for CA SOLVE:FTS access can be authorized to supervise (control) execution of system file transmissions. This field defines the user's system control privilege. Valid values for the subfield data are Y or N. If this field is omitted or specifies an invalid value, no system control privilege is assigned.

This field is ignored if the user ID has not been assigned CA SOLVE:FTS access by the earlier processing-of field X'0025'.

## SF X'0105'—CA SOLVE:FTS Private Function Mask

| | |
|---|---|
| FUNCTION: | Defines CA SOLVE:FTS private function mask |
| KEY: | X'0105' |
| SUBFIELD COUNT: | X'0001' |
| SUBFIELD 1 LENGTH: | X'000C' |
| SUBFIELD 1 DATA: | CL12' ' |

A user who is privileged for CA SOLVE:FTS access can be authorized to define, control, or request execution of private file transmissions. CA SOLVE:FTS provides a mechanism by which users can be restricted to a range of transmission definition names. This mechanism takes the form of a 12-character access mask.

This field defines the user's private access mask. The subfield data must be left justified and blank-padded to form 12 characters if required. Embedded blanks cause truncation. If this subfield is specified as blank, or this structured field is omitted, the mask defaults to the user ID name.

This field is ignored if the user ID has not been assigned CA SOLVE:FTS access by the earlier processing of field X'0025'.

## SF X'0106'—CA SOLVE:FTS System Function Mask

| | |
|---|---|
| FUNCTION: | Defines CA SOLVE:FTS system function mask |
| KEY: | X'0106' |
| SUBFIELD COUNT: | X'0001' |
| SUBFIELD 1 LENGTH: | X1000C' |
| SUBFIELD 1 DATA: | CL12' ' |

A user who is privileged for CA SOLVE:FTS access can be authorized to define, control, or request execution of system file transmissions. CA SOLVE:FTS provides a mechanism by which users can be restricted to a range of transmission definition names. This mechanism takes the form of a 12-character access mask.

This field defines the user's system access mask. The subfield data must be left justified and blank padded to form 12 characters if required. Embedded blanks cause truncation. If this subfield is specified as blank, or this structured field is omitted, the mask defaults to the string C'*'.

This field is ignored if the user ID has not been assigned CA SOLVE:FTS access by the earlier processing of field X'0025'.

## SF X‘0150’—NEWS Reset Privilege

| | |
|---|---|
| FUNCTION: | Defines CA NetMaster Network Management for SNA NEWS reset privilege |
| KEY: | X‘0150’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

A user who has NEWS access can be authorized to reset (delete) database records. Valid values are Y or N. The default is N.

Specifying Y in this field enables the user to reset statistics within the Summary Statistics Subsystem, and results in the &NEWSRSET system variable being set to YES when tested within an NCL procedure invoked by the user.

This field is ignored if the user ID has not been given access privileges to Network Management and to NEWS, that is, if the user ID does not have structured field keys X‘0022’ and X‘0026’.

## SF X‘0151’—NTS Access Privilege

| | |
|---|---|
| FUNCTION: | Defines CA NetMaster Network Management for SNA NTS access privilege |
| KEY: | X‘0151’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether a user can access the facilities of the Network Tracking System. Valid values are Y or N. The default is N.

This field is ignored if NTS is not configured on your system.

This field is ignored if the user ID has not been given access privileges to Network Management, that is, if the user ID does not have structured field key X‘0022’.

## SF X‘0180’—AOM Message Delivery and Routing Codes

| | |
|---|---|
| FUNCTION: | Defines user’s AOM message delivery and routing codes |
| KEY: | X‘0180’ |
| SUBFIELD COUNT: | X‘0002’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0010’ or X’0002’ |
| SUBFIELD 2 DATA: | BL16‘ ’ |

This field defines whether a user ID is entitled to receive AOM messages.

Subfield 1 indicates if the user is entitled to receive AOM messages while in OCS mode. The value must be Y or N. Invalid values are treated as N.

Subfield 2 is a 128-bit mask that indicates the routing codes for which the user is entitled to receive messages. The mask contains 128 bits, which are used to represent routing codes 1 to 128 inclusive, from left to right. All 16 bytes are supported, providing routing codes 1 to 128.

For example, code the value of subfield 2 as X’8200000000000000’ to indicate the user is authorized for routing codes 1 and 7—the binary equivalent of X’8200000000000000’ is B’10000010...0000’, which has the 1-bit on for routing codes 1 and 7 (reading from left to right).

## SF X‘0181’—AOM MVS SYSCMD Console Authority

| | |
|---|---|
| FUNCTION: | Defines user’s AOM SYSCMD console authority for z/OS systems |
| KEY: | X‘0181’ |
| SUBFIELD COUNT: | X‘0004’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0001’ |
| SUBFIELD 2 DATA: | CL1‘ ’ |
| SUBFIELD 3 LENGTH: | X‘0001’ |
| SUBFIELD 3 DATA: | CL1‘ ’ |
| SUBFIELD 4 LENGTH: | X‘0001’ |
| SUBFIELD 4 DATA: | CL1‘ ’ |

This field defines a user ID’s console and command authority for AOM SYSCMD on z/OS systems.

The following is a description for each of the subfields of this structured field:

- SUBFIELD 1—AOM MVS SYSCMD Console Authority level. The only values for the subfield are:

  I Information only console

  M Master console

  P Pseudo-master console

  C Subfields 2 to 4 indicate the console command authority for this user ID

Any other values in this subfield are ignored.

- SUBFIELD 2—authority to issue SYSTEM commands
- SUBFIELD 3—authority to issue IO commands
- SUBFIELD 4—authority to issue CONSOLE commands

For subfields 2 to 4 you can only enter Y or N. N is the default if a field is omitted or invalid.

## SF X‘0182’—AOM MSG Level

| | |
|---|---|
| FUNCTION: | Defines user’s AOM MSG level |
| KEY: | X‘0182’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | B‘*xxxxxx*00’ |

This field defines the 8-bit mask describing the user ID’s MSG level. Valid bit settings are as follows:

| | |
|---|---|
| 1... .... | Receive WTORs |
| .1.. .... | Receive Immediate Action |
| ..1. .... | Receive Critical Eventual Action |
| ...1 .... | Receive Eventual Action |
| .... 1... | Receive Informational |
| .... .1.. | Receive Console Broadcasts |
| .... ..11 | Reserved |

## SF X‘0183’—AOM z/VM SYSCMD Authority

| | |
|---|---|
| FUNCTION: | Defines user’s AOM SYSCMD authority for z/VM systems |
| KEY: | X‘0183’ |
| SUBFIELD COUNT: | X‘0002’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0001’ |
| SUBFIELD 2 DATA: | CL1‘ ’ |

This field defines a user ID’s AOM SYSCMD authority on z/VM systems.

The first subfield controls whether the user is authorized to issue SYSCMD DEST=PROP. Valid values are Y or N. Invalid values are treated as N.

The second subfield controls whether the user is authorized to issue SYSCMD DEST=GCS. Valid values are Y or N. Invalid values are treated as N.

This structured field is ignored by AOM on z/OS systems. It allows shared security profile for z/OS and z/VM systems.

## SF X‘0185’—AOM VOS3/JSS4 SYSCMD Command Authority

| | |
|---|---|
| FUNCTION: | Defines user’s AOM SYSCMD command authority for VOS3/JSS4 systems |
| KEY: | X‘0185’ |
| SUBFIELD COUNT: | X‘000’ |
| SUBFIELD 1 LENGTH: | X‘0001’ or X‘0002’ |
| SUBFIELD 1 DATA: | CL1‘*n* ’ or CL2‘*nn*’ |

This field provides the VOS3/JSS4 command authority level for SYSCMD. It is ignored by AOM when not in a VOS3/JSS4 environment. This allows shared security profile across all systems.

**Note:** The UAMS panel that displays AOM SYSCMD authority always shows this field and allows update. The field is not used by SYSCMD unless the system is actually VOS3/JSS4.

## SF X‘0200’—MAI-FS Privilege Class

| | |
|---|---|
| FUNCTION: | Defines user’s CA SOLVE:Access Session Management MAI-FS privilege class |
| KEY: | X‘0200’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines the MAI-FS privilege class for the user ID. Valid values are A, B, C, or D. If this field is omitted, the default is D (at user logon).

## SF X‘0201’—MAI-FS Model User ID

| | |
|---|---|
| FUNCTION: | Defines user’s SOLVE:Access Session Management MAI-FS model user ID |
| KEY: | X‘0201’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |

This field supplies an MAI-FS user ID on whose stored session definitions this user ID is to be modeled. If this field is omitted, the field is left blank.

## SF X‘0202’—MAI-FS A and E Command Capability

| | |
|---|---|
| FUNCTION: | Defines user’s CA SOLVE:Access Session Management MAI-FS A and E command capability |
| KEY: | X‘0202’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field specifies whether a user is entitled to use the A and E commands. Valid values are Y and N. If this field is omitted, the default is N.

## SF X‘0203’—MAI-FS Active Session Limit

| | |
|---|---|
| FUNCTION: | Defines user’s CA SOLVE:Access Session Management MAI-FS Active Session Limit |
| KEY: | X‘0203’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL3‘ ’ |

This field determines whether a user is limited in the number of sessions that they can have active at the same time. Possible values are 0–255. A value of 0 indicates no limit applies. This is the default when this field is omitted.

## SF X‘0204’—MAI-FS MSDM Access

| | |
|---|---|
| FUNCTION: | Defines user’s CA SOLVE:Access MAI Stored Definition Maintenance (MSDM) function access |
| KEY: | X‘0204’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field determines the level of access to the MSDM function for an A class user. The following values are possible:

- 0 specifies that the user can only update their own definitions.
- 1 specifies that the user can browse all definitions but can only update their own definitions.
- 2 specifies that the user can update all definitions.
- 3 specifies that the user can update or delete any definition.

**Default:** The command authority level determines access:

- A level less than 2 permits the user only to update their own definitions.
- A level of 2 permits the user to browse all definitions but only to update their own definitions.
- A level of 3 permits the user to update all definitions.
- A level of 4 permits the user to update or delete any definition.

## SF X‘0500’—PSM Primary Menu Access

| | |
|---|---|
| FUNCTION: | Defines user’s PSM primary menu access |
| KEY: | X‘0500’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field indicates whether the user is authorized to access the PSM : Primary Menu. Valid values are 0 (not authorized) and 1 (authorized). If this field is omitted, the default is 1 (authorized).

## SF X‘0501’—PSM Maintenance Access

| | |
|---|---|
| FUNCTION: | Defines user’s PSM maintenance access |
| KEY: | X‘0501’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines the level of access a user has to PSM printer, form, and setup definitions, and to default printer assignment. Valid values are 0 (not authorized), 1 (browse access only), 2 (browse, add, and update access), and 3 (browse, add, update, and delete access). If this field is omitted, the default is 1 (browse).

## SF X‘0502’—PSM Administration Access

| | |
|---|---|
| FUNCTION: | Defines user’s PSM administration access |
| KEY: | X‘0502’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines the level of access a user has to PSM administration functions. Valid values are 0 (not authorized), 1 (browse, and update default definitions), 2 (browse and update default definitions, and clear the spool). If this field is omitted, the default is 0 (not authorized).

## SF X‘0503’—PSM Ability to Change Print Request Priority

| | |
|---|---|
| FUNCTION: | Defines user’s PSM ability to change print request priority |
| KEY: | X‘0503’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines a user’s ability to change the priority for a PSM print request. Valid values are 0 (not authorized), and 1 (authorized). If this field is omitted, the default is 0 (not authorized).

## SF X‘0504’—PSM Queue Access for All Print Output

| | |
|---|---|
| FUNCTION: | Defines user’s PSM queue access for all print output |
| KEY: | X‘0504’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines a user’s queue access to other user’s PSM print output. Valid values are 0 (not authorized to view the output queue for other users), 1 (authorized to view the output queue for other users), 2 (authorized to browse other user output), 3 (authorized to browse, modify, and release other user output), 4 (authorized to browse, modify, release, hold, and delete other user output). If this field is omitted, the default is 1.

## SF X‘0505’—PSM Queue Access for Their Own Print Output

| | |
|---|---|
| FUNCTION: | Defines user’s PSM queue access for their own print output |
| KEY: | X‘0505’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines the user’s queue access to their own PSM print output. Valid values are 0 (not authorized to view their output queue), 1 (authorized to view their output queue), 2 (authorized to browse their output), 3 (authorized to browse, modify, and release their output), 4 (authorized to browse, modify, release, hold, and delete their output). If this field is omitted, the default is 4.

## SF X‘0510’—Panel Command Access Authority

| | |
|---|---|
| FUNCTION: | Defines user’s panel command access authority |
| KEY: | X‘0510’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether a user is entitled to enter EXEC or CMD in the COMMAND ===> and Select Option ===> input fields on panels. Valid values are Y and N. The default is N. If the user has OCS authority, this field is automatically set to Y and cannot be altered.

## SF X‘0511’—System Services Access

| | |
|---|---|
| FUNCTION: | Defines user’s security and system services access |
| KEY: | X‘0511’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field indicates whether the user has access to security and system services functions (that is, UAMS, Broadcast Services, System Support Services). Valid values are Y and N. The default is N.

## SF X‘0520’—Notification Details (First Rule)

| | |
|---|---|
| FUNCTION: | Defines user’s notification details (first rule) |
| KEY: | X‘0520’ |
| SUBFIELD COUNT: | X‘0008’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0004’ |
| SUBFIELD 2 DATA: | CL4‘ ’ |
| SUBFIELD 3 LENGTH: | X‘0007’ |
| SUBFIELD 3 DATA: | CL7‘ ’ |
| SUBFIELD 4 LENGTH: | X‘0004’ |
| SUBFIELD 4 DATA: | CL4‘ ’ |
| SUBFIELD 5 LENGTH: | X‘0004’ |
| SUBFIELD 5 DATA: | CL4‘ ’ |
| SUBFIELD 6 LENGTH: | X‘0020’ |
| SUBFIELD 6 DATA: | CL32‘ ’ |
| SUBFIELD 7 LENGTH: | X‘0008’ |
| SUBFIELD 7 DATA: | CL8‘ ’ |
| SUBFIELD 8 LENGTH: | X‘0040’ |
| SUBFIELD 8 DATA: | CL64‘ ’ |

This field defines the mode or method by which the user wishes to be notified of an event that the user has been nominated to receive. These notification details are used by various applications. Broadcast services also supports the use of the UAMS Notification Details when sending a broadcast to a specific user via the N option.

This structured field has multiple subfields containing the following information:

SUBFIELD 1—Notification Mode
SUBFIELD 2—Notification Domain
SUBFIELD 3—Day of Notification
SUBFIELD 4—Low Time
SUBFIELD 5—High Time
SUBFIELD 6—User ID
SUBFIELD 7—Exit Name
SUBFIELD 8—Parameters

If omitted, the fields are left blank.

## SF X‘0521’—Notification Details (Second Rule)

| | |
|---|---|
| FUNCTION: | Defines user’s notification details (second rule) |
| KEY: | X‘0521’ |
| SUBFIELD COUNT: | X‘0008’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0004’ |
| SUBFIELD 2 DATA: | CL4‘ ’ |
| SUBFIELD 3 LENGTH: | X‘0007’ |
| SUBFIELD 3 DATA: | CL7‘ ’ |
| SUBFIELD 4 LENGTH: | X‘0004’ |
| SUBFIELD 4 DATA: | CL4‘ ’ |
| SUBFIELD 5 LENGTH: | X‘0004’ |
| SUBFIELD 5 DATA: | CL4‘ ’ |
| SUBFIELD 6 LENGTH: | X‘0020’ |
| SUBFIELD 6 DATA: | CL32‘ ’ |
| SUBFIELD 7 LENGTH: | X‘0008’ |
| SUBFIELD 7 DATA: | CL8‘ ’ |
| SUBFIELD 8 LENGTH: | X‘0040’ |

| SUBFIELD 8 DATA: | CL64‘ ’ |
|---|---|

This field defines the mode or method by which the user wishes to be notified of an event that the user has been nominated to receive. These notification details are used by various applications. Broadcast services also supports the use of the UAMS Notification Details when sending a broadcast to a specific user via the N option.

This structured field has multiple subfields containing the following information:

SUBFIELD 1—Notification Mode
SUBFIELD 2—Notification Domain
SUBFIELD 3—Day of Notification
SUBFIELD 4—Low Time
SUBFIELD 5—High Time
SUBFIELD 6—User ID
SUBFIELD 7—Exit Name
SUBFIELD 8—Parameters

If omitted, the fields are left blank.

## SF X‘0522’—Notification Details (Third Rule)

| | |
|---|---|
| FUNCTION: | Defines user’s notification details (third rule) |
| KEY: | X‘0522’ |
| SUBFIELD COUNT: | X‘0008’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0004’ |
| SUBFIELD 2 DATA: | CL4‘ ’ |
| SUBFIELD 3 LENGTH: | X‘0007’ |
| SUBFIELD 3 DATA: | CL7‘ ’ |
| SUBFIELD 4 LENGTH: | X‘0004’ |
| SUBFIELD 4 DATA: | CL4‘ ’ |
| SUBFIELD 5 LENGTH: | X‘0004’ |
| SUBFIELD 5 DATA: | CL4‘ ’ |
| SUBFIELD 6 LENGTH: | X‘0020’ |
| SUBFIELD 6 DATA: | CL32‘ ’ |
| SUBFIELD 7 LENGTH: | X‘0008’ |
| SUBFIELD 7 DATA: | CL8‘ ’ |
| SUBFIELD 8 LENGTH: | X‘0040’ |
| SUBFIELD 8 DATA: | CL64‘ ’ |

This field defines the mode or method by which the user wishes to be notified of an event that the user has been nominated to receive. These notification details are used by various applications. Broadcast services also supports the use of the UAMS Notification Details when sending a broadcast to a specific user via the N option.

This structured field has multiple subfields containing the following information:

SUBFIELD 1—Notification Mode
SUBFIELD 2—Notification Domain
SUBFIELD 3—Day of Notification
SUBFIELD 4—Low Time
SUBFIELD 5—High Time
SUBFIELD 6—User ID
SUBFIELD 7—Exit Name
SUBFIELD 8—Parameters

If omitted, the fields are left blank.

## SF X‘0523’—Notification Details (Fourth Rule)

| | |
|---|---|
| FUNCTION: | Defines user’s notification details (fourth rule) |
| KEY: | X‘0523’ |
| SUBFIELD COUNT: | X‘0008’ |
| SUBFIELD 1 LENGTH: | X‘0008’ |
| SUBFIELD 1 DATA: | CL8‘ ’ |
| SUBFIELD 2 LENGTH: | X‘0004’ |
| SUBFIELD 2 DATA: | CL4‘ ’ |
| SUBFIELD 3 LENGTH: | X‘0007’ |
| SUBFIELD 3 DATA: | CL7‘ ’ |
| SUBFIELD 4 LENGTH: | X‘0004’ |
| SUBFIELD 4 DATA: | CL4‘ ’ |
| SUBFIELD 5 LENGTH: | X‘0004’ |
| SUBFIELD 5 DATA: | CL4‘ ’ |
| SUBFIELD 6 LENGTH: | X‘0020’ |
| SUBFIELD 6 DATA: | CL32‘ ’ |
| SUBFIELD 7 LENGTH: | X‘0008’ |
| SUBFIELD 7 DATA: | CL8‘ ’ |
| SUBFIELD 8 LENGTH: | X‘0040’ |

| | |
|---|---|
| SUBFIELD 8 DATA: | CL64‘ ’ |

This field defines the mode or method by which the user wishes to be notified of an event that the user has been nominated to receive. These notification details are used by various applications. Broadcast services also supports the use of the UAMS Notification Details when sending a broadcast to a specific user via the N option.

This structured field has multiple subfields containing the following information:

SUBFIELD 1—Notification Mode
SUBFIELD 2—Notification Domain
SUBFIELD 3—Day of Notification
SUBFIELD 4—Low Time
SUBFIELD 5—High Time
SUBFIELD 6—User ID
SUBFIELD 7—Exit Name
SUBFIELD 8—Parameters

If omitted, the fields are left blank.

## SF X‘0530’—TCP/IP Services Access Privilege

| | |
|---|---|
| FUNCTION: | Defines a user’s TCP/IP Services authority level. |
| KEY: | X‘0530’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

Defines the authority level for a user using TCP/IP Services. Valid values are 0 (not authorized to access TCP/IP Services), 1 (browse only authority), and 2 (full update authority which is necessary to: run Obeyfiles, drop connections, perform packet tracing, start and stop devices, and use the SNMP Set option). If omitted, the default is 0.

This field is ignored if the user ID has not been given access privileges to Network Management, that is, if the user ID does not have structured field key X‘0022’.

## SF X‘0550’—Report Writer Primary Menu Access

| | |
|---|---|
| FUNCTION: | Defines user’s Report Writer primary menu access |
| KEY: | X‘0550’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether a user ID is authorized to access the Report Writer primary menu. Valid values are 0 (not authorized), and 1 (authorized). If omitted, the default is 1.

## SF X‘0551’—Report Writer Administration Access

| | |
|---|---|
| FUNCTION: | Defines user’s Report Writer administration access |
| KEY: | X‘0551’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether a user ID is authorized to access Report Writer table maintenance administration functions. Valid values are 0 (not authorized), and 1 (authorized). If omitted, the default is 0.

## SF X‘0552’—Report Writer Maintenance Access

| | |
|---|---|
| FUNCTION: | Defines user’s Report Writer maintenance access |
| KEY: | X‘0552’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether a user is authorized to access the Report Writer report definition menu. Valid values are 0 (not authorized for menu access), and 1 (menu access authorized). If omitted, the default is 1.

## SF X‘0553’—Report Writer Public Report Access

| | |
|---|---|
| FUNCTION: | Defines user’s Report Writer public report access |
| KEY: | X‘0553’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines the level of access a user ID has to public reports. Valid values are 0 (not authorized), 1 (authorized to generate, browse, and copy), 2 (authorized to generate, browse, copy, add, and update), 3 (authorized to generate, browse, copy, add, update, and delete). If omitted, the default is 1.

## SF X‘0554’—Report Writer Access to Their Own Reports

| | |
|---|---|
| FUNCTION: | Defines user’s Report Writer access to their own reports |
| KEY: | X‘0554’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines the level of access a user ID has to their own reports. Valid values are 0 (not authorized), 1 (authorized to generate, browse, and copy), 2 (authorized to generate, browse, copy, add, and update), 3 (authorized to generate, browse, copy, add, update, and delete). If omitted, the default is 3.

## SF X‘0555’—Report Writer Private Report Access for All Users

| | |
|---|---|
| FUNCTION: | Defines user’s Report Writer private report access for all users |
| KEY: | X‘0555’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines the level of access a user ID has to private reports for all users. Valid values are 0 (not authorized), 1 (authorized to generate, browse, and copy), 2 (authorized to generate, browse, copy, add, and update), 3 (authorized to generate, browse, copy, add, update, and delete). If omitted, the default is 0.

## SF X‘0556’—Report Writer Schedule Maintenance Access

| | |
|---|---|
| FUNCTION: | Defines user’s Report Writer schedule maintenance access |
| KEY: | X‘0556’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines the level of access a user ID has to Report Writer schedule definitions. Valid values are 0 (not authorized), 1 (authorized to browse), 2 (authorized to browse, add, and update), 3 (authorized to generate, browse, add, update, and delete), and 4 (authorized for the Stop and Start functions of schedule processing). If omitted, the default is 1.

## SF X‘0580’—Access to SOLVE:NetMail

| | |
|---|---|
| FUNCTION: | Defines user’s access to CA SOLVE:NetMail (Electronic Mail field) |
| KEY: | X‘0580’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether a user ID has access to CA SOLVE:NetMail. Valid values are Y or N. If omitted, the default is N.

## SF X‘0601’—Access to Managed Objects Development Services (MODS)

| | |
|---|---|
| FUNCTION: | Defines user’s access to Managed Objects Development Services (MODS) |
| KEY: | X‘0601’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether a user ID has access to MODS. Valid values are Y or N. If omitted, the default is N.

## SF X‘0605’—Object Services Access

| | |
|---|---|
| FUNCTION: | Defines user’s access to Object Services |
| KEY: | X‘0605’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether a user ID has access to Object Services support functions. Valid values are Y or N. If omitted, the default is N.

## SF X‘0609’—Object Services Security Access

| | |
|---|---|
| FUNCTION: | Defines user’s security access to Object Services |
| KEY: | X‘0609’ |
| SUBFIELD COUNT: | X‘0001’ |
| SUBFIELD 1 LENGTH: | X‘0001’ |
| SUBFIELD 1 DATA: | CL1‘ ’ |

This field defines whether a user ID has access to Object Services. Valid values are Y or N. If omitted, the default is N.

# Appendix C: User ID Security Exit Support

This section contains the following topics:

## External Security Packages

External security packages such as CA ACF2 or RACF can provide minimal or full security checking.

If your organization has an external security package, access to that package is provided through one of the following types of exit:

- Partial security exit—password and logon access maintenance is controlled by the external security package while UAMS stores the user definitions.
- Full security exit—all security functions are maintained and stored by your external security package.

## Sample Exits

The following sample security exits are provided:

**CCACF2FX**

Is a full security exit for CA ACF2.

**CCRACFFX**

Is a full security exit for RACF.

**NMSAFPX**

Is a SAF partial security exit.

Sample exits are in the following libraries:

- The SMP target zone library, *dsnpref*.*pvpref*.CC2DSAMP, where:
  - *dsnpref* is your site-specific data set name prefix
  - *pvpref* is your product version prefix
  - CC2DSAMP is the data set for all products
- The SMP distribution zone library, *dsnpref*.*pvpref*.AC2SAMP, where:
  - *dsnpref* is your site-specific data set name prefix.
  - *pvpref* is your product version prefix.
  - AC2DSAMP is the data set for all products.

**Note:** The SMP target zone is updated when you SMP APPLY maintenance, whereas the SMP distribution zone is updated only when you subsequently SMP ACCEPT the maintenance. To ensure that you include all applied maintenance, it is recommended that you use the member in the SMP target zone (*dsnpref*.*pvpref*.CC2DSAMP).

**Note:** On z/VM systems, these exits are on the *vmid* 193 C-disk.

### NMSAFPX Partial Security Exit

The NMSAFPX partial security exit is a SAF-based security exit that supports UTOKENS. SAF is the IBM System Authorization Facility and is the agreed standard for the encoding of requests that require security checking.

**Note:** SAF is documented in IBM's *Security Server RACROUTE Macro Reference* manual. See the documentation for your security package to find out whether the package supports SAF-formatted calls.

You can make a copy of NMSAFPX and change it to suit your requirements. To assemble and link your exit, use the sample JCL member NMSAFPXL, which is in the same data set as the NMSAFPX sample exit.

The load module created by the link-edit step should be placed in the load library, or in another library concatenated to the load library through the STEPLIB DD statement in the started task JCL. Change the JCL parameters to include SEC=*name*, where *name* identifies the load module. This causes the security exit to be used.

**Note:** The supplied SEC=PARTSAF option is functionally equivalent to the NMSAFPX sample.

## How You Write Your Own User ID Security Exit

The security exit is coded as an assembler language module, or suite of modules. These modules must be able to:

- Accept the various parameter lists passed to it
- Return the designated return codes associated with those parameter lists

When the exit has been written, you link it into a standalone load module. The name of this load module is arbitrary (for example, MYEXIT). The load module is placed in an APF authorized library or another library concatenated to the load library in the JCL STEPLIB DD statement. Change the JCL parameters to include SEC=*name*, where *name* identifies the load module (for example, SEC=MYEXIT).

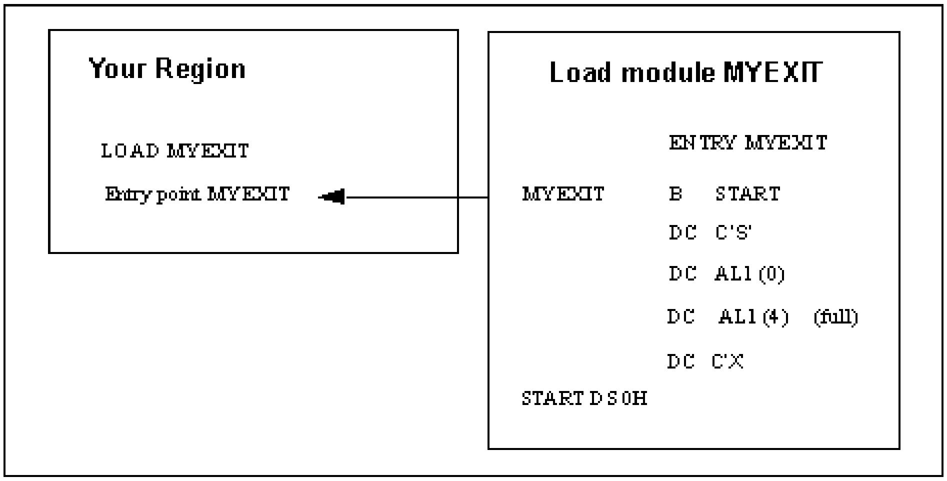
Your region determines the type (partial or full security) of exit being provided by examining the data at the entry point of the exit load module.

The entry point of the security exit load module must look like the following code:

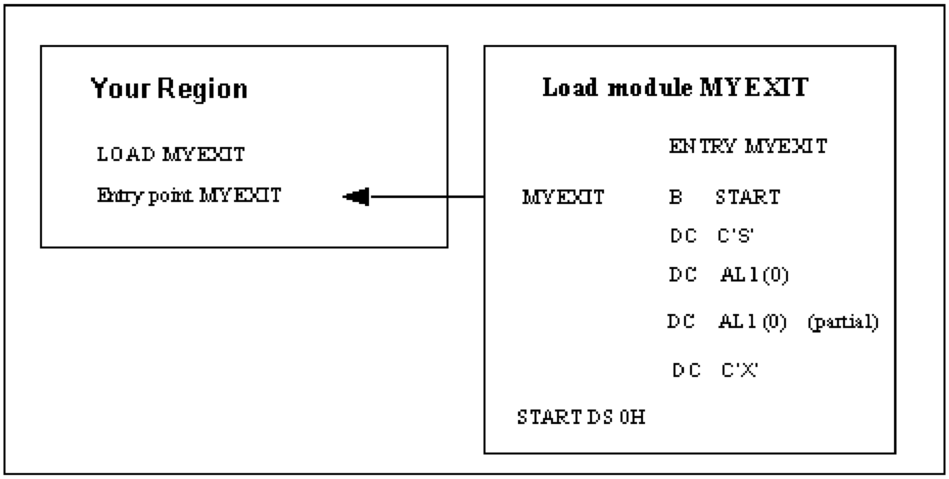
```
ENTRY  B    SKIP-ENTRY(,R15)   Skip descriptor
       DC   C'S'               Required, constant 'S'
       DC   AL1 (0)            Required, interface version,
       DC   AL1 (type)         Required, exit type,
                                         0: partial
                                         4: full
       DC   C'X'               Required, constant 'X'
```

The security exit can be AMODE 24 or 31, and RMODE 24 or ANY. All data areas and parameter lists passed to it are located in storage below 16M.

This diagram shows how a region loads and identifies a full security exit and how your region locates the entry point of the exit's code.



This diagram shows how a region loads and identifies a partial security exit and how your region locates the entry point of the exit's code.



```
LOAD MYEXIT
Entry point MYEXIT
```

```
              ENTRY MYEXIT
MYEXIT        B    START
              DC   C'S'
              DC   AL1(0)
              DC   AL1(0)   (partial)
              DC   C'X'
START DS 0H
```

## Exit Execution

The exit executes within an operating system subtask and can therefore issue WAITs or SVCs that suspend the task without affecting the primary task.

**Note:** In z/OS systems, the subtask is attached sharing subpool 50 with the main task. This allows other exits (which also use subpool 50) to share common storage areas with the security exit.

Calls to the exit are serialized across the system—if one call to the exit is in progress, any subsequent calls are queued and processed one-by-one, in request order. Serial request processing means that the exit does not have to be written as reentrant.

The exit is called for two reasons:

- Information—to keep the security system informed of developments or system changes
- Function request—to request that a function be performed

**Important!** If an abend occurs in the exit and the requested function cannot be performed it is regarded as a security exposure and the region terminates with the internal abend 268-01.

## Supported Exit Calls

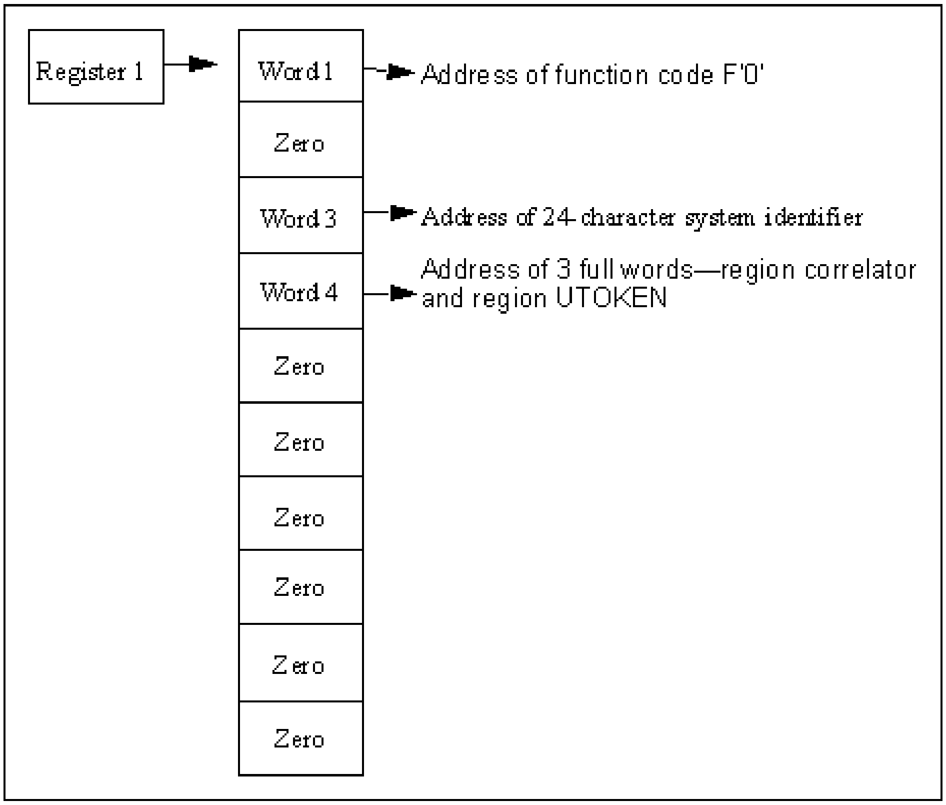
The following table lists the calls supported by the exit, their type, and their function codes:

| Call | Type | Function Code |
|---|---|---|
| System Initialization | information | 0 |
| System Close Down | information | 4 |
| User Logon Request | function request | 8 |
| User Logoff Request | information | 12 |
| Logon Verification | function request | 16 |
| Change Password | function request | 20 |
| Return User ID Information | function request | 24 |
| Update User ID | function request | 28 |
| &SECCALL EXIT | function request | 32 |
| Return Sequential User ID Information | function request | 36 |
| Add User ID | function request | 40 |
| Delete User ID | function request | 44 |

The following sections provide the parameter lists for each call.

# System Initialization Parameter List

On entry to the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'0'
Zero
Word 3
Address of 24-character system identifier
Word 4
Address of 3 full words—region correlator and region UTOKEN
Zero
Zero
Zero
Zero
Zero
Zero

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘0’.

**Word 2**

Is set to zero.

**Word 3**

Is a 24-character area containing the following:

- The eight-character primary ACB name field (padded to the right with blanks if necessary). The primary ACB name is as specified on the PRI JCL initialization parameter.
- Four blanks
- The four-character system domain ID as coded on the NMDID initialization parameter. If the NMDID parameter is omitted, the domain ID defaults to one of the following values:
  - The first four characters of the primary ACB name used by your region (blank padded if necessary)
  - The ACB name itself if less than five characters long.
- The four-character system user prefix as coded on the NMSUP initialization parameter. If not coded, the value defaults to the NMDID value.
- The four-character product version, for example, V5.1.

These characters are passed to the exit to identify the region that is executing and can be regarded as the user ID of the system. The exit can use this value to determine whether, for example, it is a production region.

**Word 4**

Is the address of two full words:

- First word can contain any value. This value is returned in the system closedown call, so it can be used as a region correlator. The exit can use this word to relate this region to a set of control blocks or other information maintained by the exit in relation to the region.
- Second word is the region user token (UTOKEN) if one is available.

**Words 5 through 10**

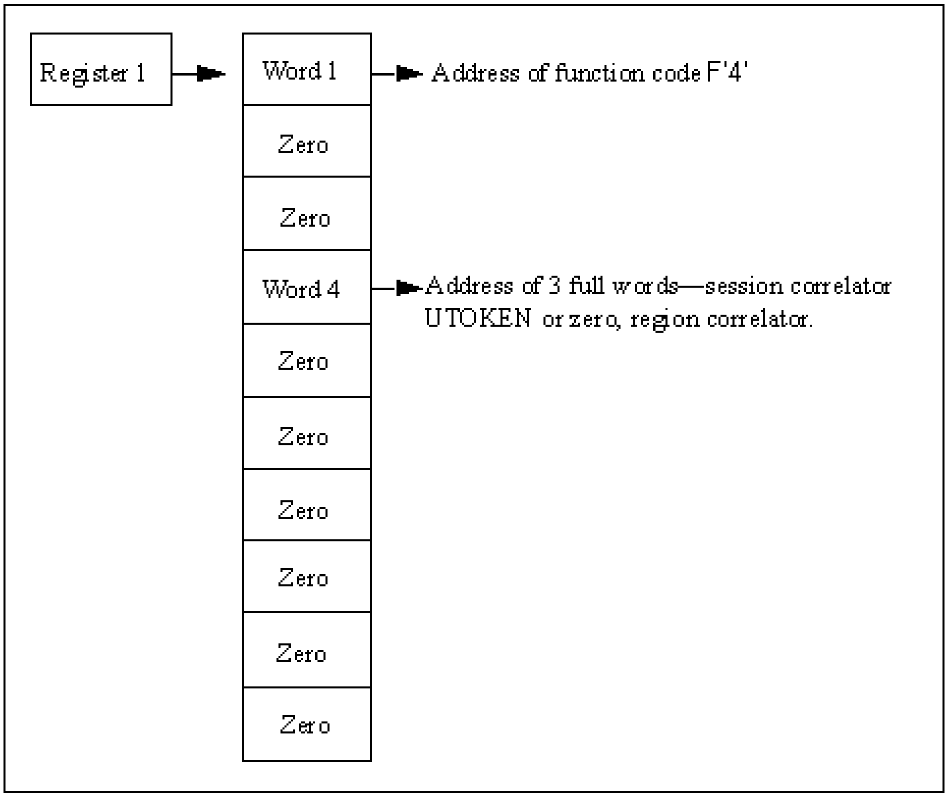
Are set to zeros.

## Return Codes from Initialization Call

The exit must return a completion code in Register 15 on return. Successful initialization must be signified by return code F‘0’. Your product region terminates if any other value is returned in Register 15 on completion of this call.

# System Close Down Parameter List

On entry to the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'4'
Zero
Zero
Word 4
Address of 3 full words—session correlator
UTOKEN or zero, region correlator.
Zero
Zero
Zero
Zero
Zero
Zero

The contents of the parameter list as follows:

**Word 1**

Is a fullword containing the address of function code F‘4’.

**Words 2 and 3**

Are set to zeros.

**Word 4**

Is the address of three full words, each containing the information that was passed on the system initialization call.

**Words 5 through 10**

Are set to zeros.

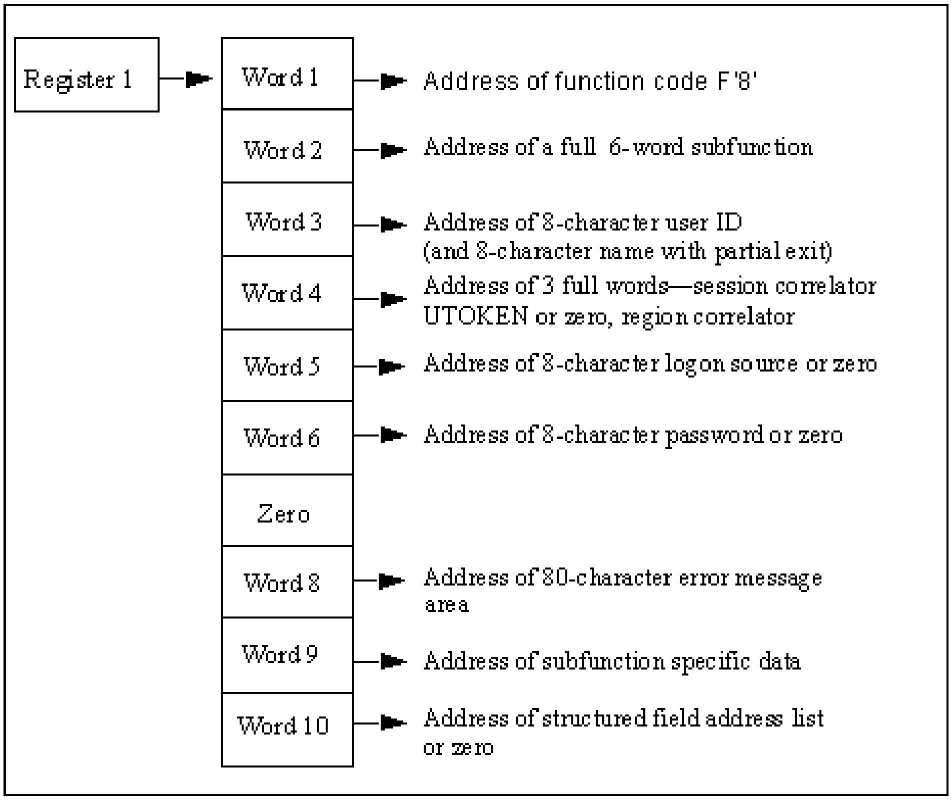
The processing associated with this call depends upon the environment maintained by the exit, but can include closing any data sets used by the exit.

## Return Codes from Closedown Call

The exit must return a completion code in Register 15 on return. Successful processing of the close down call must be signified by return code F‘0’.

# Logon Request Parameter List

On entry to the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'8'
Word 2
Address of a full 6-word subfunction
Word 3
Address of 8-character user ID
(and 8-character name with partial exit)
Word 4
Address of 3 full words—session correlator
UTOKEN or zero, region correlator
Word 5
Address of 8-character logon source or zero
Word 6
Address of 8-character password or zero
Zero
Word 8
Address of 80-character error message area
Word 9
Address of subfunction specific data
Word 10
Address of structured field address list or zero

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘8’.

### Word 2

Is the address of a subfunction code. This code defines the type of logon request being processed and has values set as follows:

| Subfunction Code | Value |
|---|---|
| F‘0’ | Logon from native terminal |
| F‘4’ | Logon from TSO interface component of EIP |
| F‘8’ | Logon from ROF user |
| F‘12’ | System console logon |
| F‘16’ | System environment logon |
| F‘20’ | Reserved |
| F‘24’ | Reserved |
| F‘28’ | APPC user region logon |
| F‘32’ | Reserved |

### Word 3

Is the address of an eight-character field, blank padded, containing the user ID requesting the logon. If a partial security exit is installed then there is an additional, contiguous eight-character field, blank-padded containing the model user ID specified by the SYSPARMS MODLUSER operand. If no model user has been specified this field is set to blanks.

### Word 4

Is the address of two full words:

- First word can contain any value to relate a user to a set of control blocks or other information maintained by the exit in relation to that user.
- Second word contains the address of the user token (UTOKEN) obtained from the SAF logon call.

These values are returned in all subsequent calls to the exit in reference to this user ID session, so that it can be used as a session correlator.

### Word 5

Varies according to subfunction code.

| Subfunction Code | Value |
|---|---|
| F‘0’ | Word 5 contains the address of an eight-character field, blank-padded, which contains the name of the terminal at which the user wants to logon. |
| F‘4’ | Word 5 contains the address of an eight-character field, blank-padded, which contains the name of the VTAM APPL name used by the TSO interface for this session. |
| F‘8’ | Word 5 is set to zero. |
| F‘12’ | Word 5 contains the address of an eight-character field holding the word CONSOLE, or the generated user ID for multiple console support (OS/VS systems only). |
| F‘16’ | Word 5 contains the address of an eight-character field containing the name of the pseudo-terminal with which the system environment logon is associated. |
| F‘28’ | Word 5 is set to zero. |

### Word 6

Varies according to subfunction code.

| Subfunction Code | Value |
|---|---|
| F‘0’ | Word 6 contains the address of an eight-character field, blank-padded, containing the password entered with the user ID. |
| F‘4’ | Word 6 is set to zero. |
| F‘8’ | Word 6 contains the address of an eight-character field, blank-padded, holding the password entered on a SIGNON command from the remote region. If no password was entered, this word is zero. |
| F‘12’ | Word 6 is set to zero. |
| F‘16’ | Word 6 is set to zero. |
| F‘28’ | Word 6 contains the address of an eight-character field, blank-padded, holding the password specified on the APPC NCL verb that started this region. If no password was entered, this word is zero. |
| F’36’ | Web user logon. |

### Word 7

Is set to zero.

### Word 8

Is the address of an 82-character area into which the exit can place any error message text if the logon request is denied. Any text placed in this area is displayed to the user in response to the logon attempt. If an error message is returned in this area it must be formatted as follows:

#### Bytes 00 and 01

Specifies the hexadecimal length of the message text (excluding these two bytes).

#### Bytes 02 through *nn*

Specifies the error message text.

**Limits:** 80 characters

Message text is converted to uppercase.

### Word 9

Varies according to subfunction code.

| Subfunction Code | Value |
|---|---|
| F‘0’ | Word 9 set to zero. |
| F‘4’ | Word 9 set to zero. |
| F‘8’ | Address of 12-character area holding the INMC link name of the region from which a ROF logon request has been received. Following this address is a four-character field that is either zero or contains the domain ID of the region from which the SIGNON request originated. Following this field is a one-byte binary field with flag settings as follows:<br>■ X‘01’—A character domain ID is present after the 12-character link name.<br>■ X‘02’—Is set if the system represented by domain ID is different from the system represented by the link name. |
| F‘12’ | Word 9 set to zero. |
| F‘16’ | Word 9 set to zero. |
| F‘28’ | Word 9 set to zero. |

### Word 10

Depends on whether you have a full or partial security system:

- Full security exit—the logon is accepted and the exit must provide the address of a list of full words in this field, terminated by X‘FFFFFFFF’.

  Each word in this list contains the address of a structured field defining an item of information for this user ID. Your region processes all these structured fields to determine the privileges and other information to be associated with the user ID.

- Partial security exit—set to zero, or the address of a variable length list of full words, ended by a fullword of X‘FFFFFFFF’.

  Each word in the area points to a single structured field, representing one attribute of the current UAMS definition for the user ID pointed to by word 3 of the parameter list.

If the requesting user ID is not defined on UAMS, Word 10 is zero.

## Return Codes from Logon Call

The exit must return a completion code in Register 15 on return. Completion codes are supported as described below. Specific causes for logon rejection are identified by error message text returned in the area addressed by word 8 of the Logon Call parameter list.

These return codes are the only codes that are accepted in response to a logon call. Any other return code is treated as indicating that the logon is rejected.

**0**

Indicates that logon is accepted without error.

**4**

Indicates that logon is accepted but password has expired. Your product region must enforce password renewal before the user has access to any other functions.

**8**

Indicates that logon is accepted but this is a new user ID. Your product region must enforce password change before allowing the user access to any other functions.

**12**

Is reserved and cannot be used as a return code.

**16**

Indicates that logon is rejected. Password is incorrect, but your product region is to allow a retry.

**20**

Indicates that logon is rejected. Password is incorrect and no retry is to be permitted. A violation message is logged.

**24**

Indicates that logon is rejected. An error message explaining the cause of rejection is available in the 82-character area addressed by word 8 of the parameter list. The format of this area must be a 2-byte length field set to contain the length of the error message text, followed by up to 80 characters of error message text. If the logon was from a TSO user, the error message is not displayed; the normal product region logon panel is presented for entry of user ID and password and the user is not classified as a TSO interface user.

**Note:** Return codes 16 and 20 are provided to give compatibility with the standard UAMS functions, which provide a maximum number of password retries before rejecting the logon attempt and logging a violation message.

Return codes 4 and 8 provide compatibility with the UAMS convention of enforcing password change at logon time if the user's password has expired, or for the first logon of a new user ID.

The exit might not be able to determine that a user ID is new, depending upon the information available to it from the external security system in use. If required, the exit can be written to provide two return codes only, 0 or 24, with an appropriate error message if the logon is rejected.

The ability to deny a logon with return code 24 and an error message of the exit's choice allows the installation to extend the security exit's function beyond that of simple password validation. For example, the exit might deny logons after a certain time of day or reject ROF logons from certain remote systems.

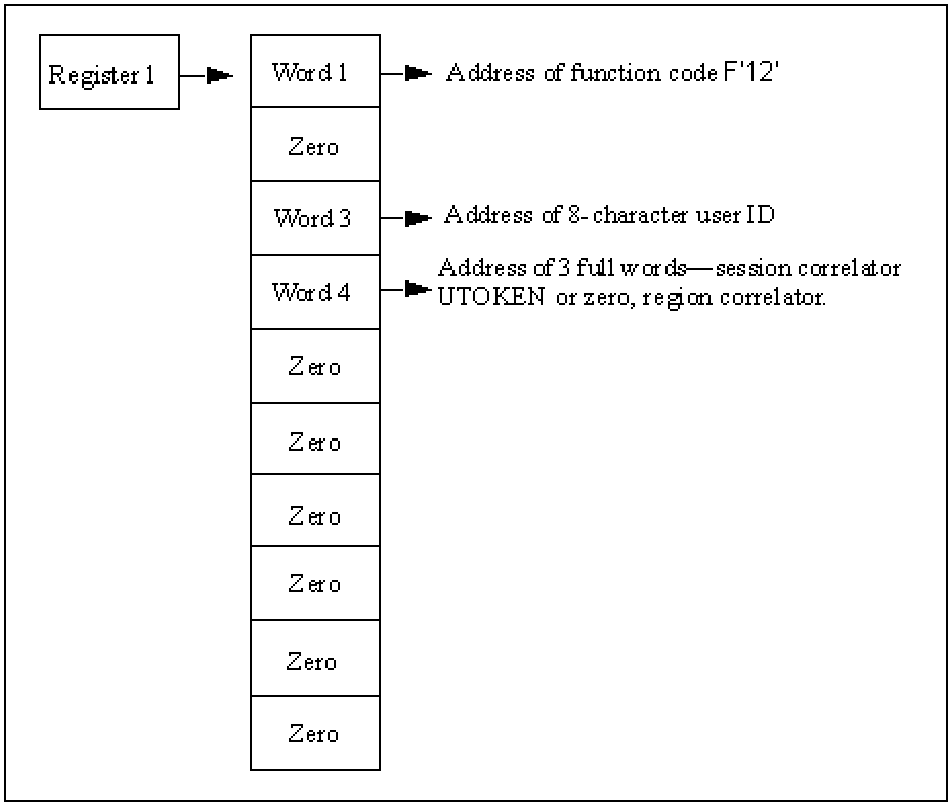
System console and system environment logon calls take default attributes if the exit causes the logon to fail.

**More information:**

Access to Your Region (see page 44)

# Logoff Request Parameter List

On entering the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'12'
Zero
Word 3
Address of 8-character user ID
Word 4
Address of 3 full words—session correlator UTOKEN or zero, region correlator.
Zero
Zero
Zero
Zero
Zero
Zero

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘12’.

**Word 2**

Is set to zero.

**Word 3**

Is the address of an eight-character field, blank-padded, containing the user ID logging off.

**Word 4**

Is the address of three full words containing the user session correlator (if any) and the user token (UTOKEN) address, set by the exit when this user ID logged on, or zero.

**Words 5 through 10**

Are set to zero.

## Return Codes from Logoff Calls

The exit must return a completion code in register 15 on return. The only valid completion code is F‘0’.

**Note:** It is possible for the exit to receive a logoff call for a user who has never successfully logged on. This might occur when a user abandons their logon attempt due to a forgotten password. The exit must be written to accept such calls and to ignore them.

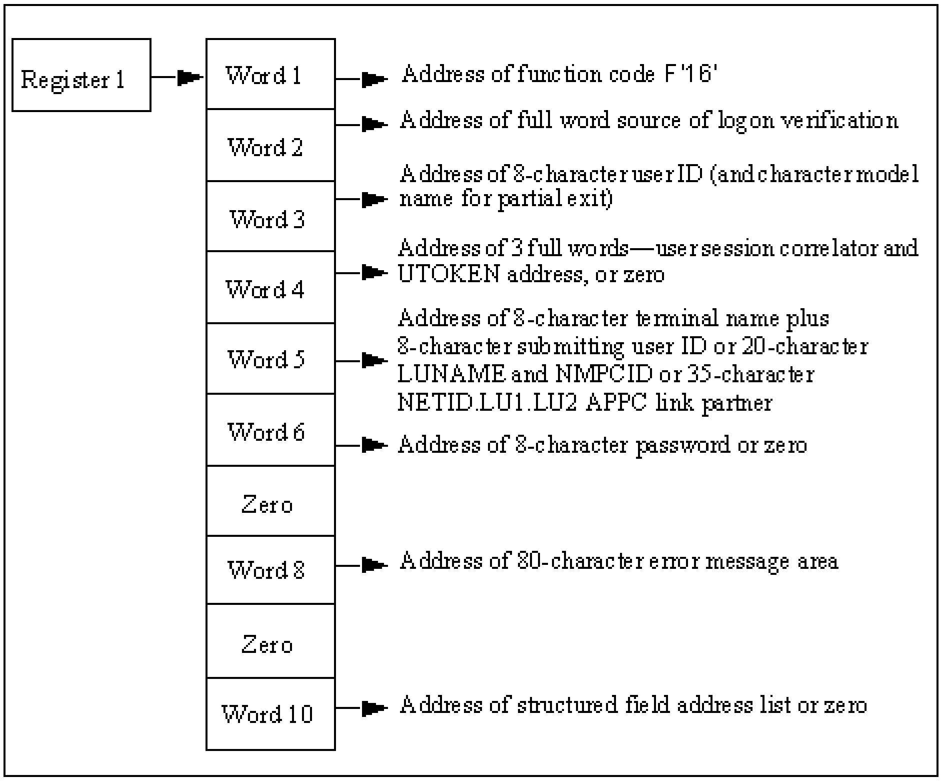
Your system services generate a logoff call during DISCONNECT processing on behalf of an internally created environment. Because there was no logon request call for this environment, there is no session correlator created, and it is, therefore, null.

When a users logs back on to reconnect, a logon request call is made to the security exit. A new session correlator is created at this time. When the user selects a RECONNECT or CANCEL of all disconnected environments, a logoff request call is made to the security exit with the original user session correlator. The reconnected or new session continues with a new user session correlator that is presented to the security exit upon that session’s logoff.

A zero user session correlator on a logoff request call to the security exit is valid for a DISCONNECT request only.

# Logon Verification Call Parameter List

On entry to the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Word 2
Word 3
Word 4
Word 5
Word 6
Zero
Word 8
Zero
Word 10
Address of function code F'16'
Address of full word source of logon verification
Address of 8-character user ID (and character model name for partial exit)
Address of 3 full words—user session correlator and UTOKEN address, or zero
Address of 8-character terminal name plus 8-character submitting user ID or 20-character LUNAME and NMPCID or 35-character NETID.LU1.LU2 APPC link partner
Address of 8-character password or zero
Address of 80-character error message area
Address of structured field address list or zero

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘16’.

**Word 2**

Is the address of a subfunction code. This code defines the source of the logon verification call and has values set as follows:

| Subfunction Code | Value |
|---|---|
| F‘0’ | Call is from an &SECCALL CHECK verb, the LOCK facility, or APPC region validation. |
| F‘4’ | Call is from INMC link activation. |
| F‘8’ | Call is from APPC link activation. |

**Word 3**

Is the address of a 16-character field, blank-padded. The field contains the eight-character user ID whose password and logon capability is to be verified. The ID is followed by the eight-character system default MODLUSER name, or blanks if no model user has been defined.

**Word 4**

Is the address of three full words containing the user session correlator and the address of the user token (UTOKEN) of the submitter of the logon verification call, or zero if called from EASINET.

**Word 5**

Varies according to subfunction code:

| **Subfunction Code** | **Value** |
|---|---|
| F‘0’ | Address of the eight-character terminal ID, followed by the eight-character user ID of the submitting user. This field can be blank; if so, treat the user ID being checked as the submitting user. |
| F‘4’ | Address of the 8-character LU name of the link, followed by the 12-character NMPCID. |
| F‘8’ | Address of the 35-character APPC link partners. The format of the link session partners is NETID.LU1.LU2 where NETID.LU1 is the source partner and LU2 is the destination LU of the APPC link. |

**Word 6**

Varies according to subfunction code:

| **Subfunction Code** | **Value** |
|---|---|
| Subfunction F‘0’ | Address of the 8-character, blank-padded user ID password. |
| Subfunction F‘4’ | Word 6 is set to zero. |
| Subfunction F‘8’ | Address of an eight-character area into which the exit can place the session key for this APPC link. |

**Word 7**

Is set to zero.

**Word 8**

Is the address of an 82-character area into which the exit can place an error message. If an error message is returned, it must be formatted as follows:

**Bytes 00 and 01**

Specifies the hexadecimal length of the message text (excluding these two bytes).

**Bytes 02 through *nn***

Specifies the error message text.

**Limits:** 80 characters

Message text is converted to uppercase. An error message might be returned in this area for return codes 20 or 24. A message set in this area for any other return code is ignored.

**Word 9**

Is set to zero.

**Word 10**

If you have a full security exit, this word is set to zero.

If you have a partial security exit this word is set to zero if the user ID is not defined on the UAMS data set. Otherwise word 10 points to a list of full words, ended by a fullword of X‘FFFFFFFF’. Each address in the list points to a single structured field representing one attribute of the user ID to verify. This list therefore provides the exit with access to the entire current definition of this user ID.

## Return Codes from Logon Verification Call

The exit must return a completion code in Register 15 on return. Completion codes are supported as described below.

The return codes listed below are for subfunction codes 0 and 4. They are the only codes that are accepted in response to a Logon Verification call. Any other return code is rejected and set to 24.

**0**

Indicates that the password is valid. Logon is successful.

**4**

Indicates that the password is valid but expired. This is not a new user ID. The user is prompted to change the password before logon is successful.

**8**

Indicates that the password is valid but this is a new user ID. The user is prompted to change the password before logon is successful.

**16**

Indicates that the password is wrong.

**20**

Indicates that the password is correct but logon is rejected. The exit might return an explanatory message in the error message area addressed by word 8 of the parameter list.

**24**

Indicates that the request failed or function is not supported by the exit. The exit might return an explanatory message in the error message area addressed by word 8 of the parameter list.

The return codes listed below are for subfunction code 8. They are the only codes that are accepted in response to an APPC link establishment Logon Verification call. Any other return code is rejected and set to 24.

**0**

Indicates that link activation is successful. The session key is addressed by word 6 of the parameter list.

**4**

Indicates that link activation is successful. No session key is available.

**8**

N/A

**16**

Indicates that link activation is unsuccessful.

**20**

N/A

**24**

Indicates that the function is not supported. The link establishment is to continue without a session key.

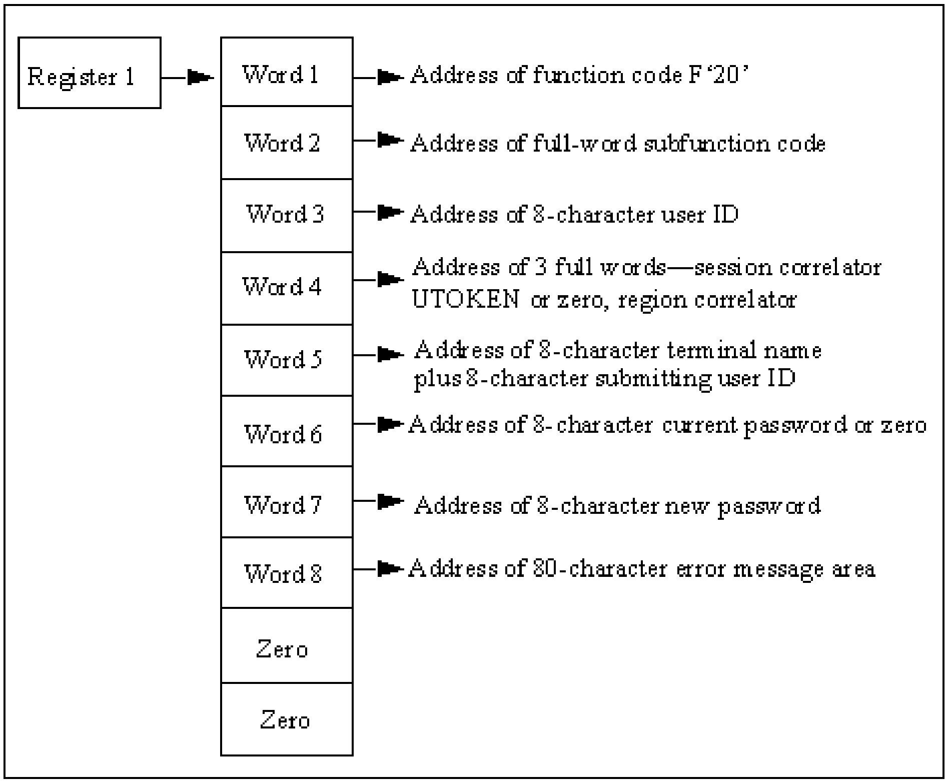
**Notes:**

- When a Logon Verification Call to the exit is made by a &SECCALL CHECK statement or by NMPC link activation, no user correlator information can be provided. Word 4 of the parameter list is set to zero.
- The Logon Verification Call return codes do not strictly correlate with the &SECCALL CHECK return code values. For an explanation of &SECCALL CHECK return codes, see the *Network Control Language Reference.*
- If the Logon Verification call is made by the LOCK facility, the call is being made on behalf of a logged-on user. Consequently, the parameter list provides the address of the appropriate user session correlator.

The Logon Verification Call is primarily a means of querying the validity of a password, and the exit can be written to provide only this function. The option of supporting return code 20 is provided to complement support of return code 24 from the Logon Request Call, which allows the exit to refuse a logon request for reasons other than password error.

# Change Password Parameter List

On entry to the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'20'
Word 2
Address of full-word subfunction code
Word 3
Address of 8-character user ID
Word 4
Address of 3 full words—session correlator UTOKEN or zero, region correlator
Word 5
Address of 8-character terminal name plus 8-character submitting user ID
Word 6
Address of 8-character current password or zero
Word 7
Address of 8-character new password
Word 8
Address of 80-character error message area
Zero
Zero

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘20’.

**Word 2**

Is the address of a subfunction code. This code defines the source of the change password request and has values set as follows:

| **Subfunction Code** | **Value** |
|---|---|
| F‘0’ | User has requested password change |
| F‘4’ | Change required because password expired |
| F‘8’ | Forced change by UAMS user |

| | |
|---|---|
| F‘12’ | &SECCALL CHANGE request issued from an EASINET procedure |

**Word 3**

Is the address of an eight-character field, blank-padded, containing the user ID whose password is to change.

**Word 4**

Varies according to subfunction code:

| Subfunction Code | Value |
|---|---|
| F‘0’ | Word 4 is the address of two full words. The first contains the correlator and the second the user token (UTOKEN) of the user issuing the change request. |
| F‘4’ | As for subfunction F‘0’. |
| F‘8’ | As for subfunction F‘0’. |
| F‘12’ | Word 4 is zero. |
| F‘16’ | As for subfunction F‘0’. |

**Word 5**

Is the address of an eight-character terminal name followed by the eight-character user ID of the user issuing the change request.

**Word 6**

Varies according to subfunction code:

| Subfunction Code | Value |
|---|---|
| F‘0’ | Word 6 contains the address of an eight-character field, blank-padded, which contains the current user ID password. |
| F‘4’ | As for subfunction F‘0’. |
| F‘8’ | Word 6 is set to zero. |
| F‘12’ | As for subfunction F‘0’. |
| F‘16’ | Web user logon. |

**Word 7**

Is the address of an eight-character field containing the new password to assign to the user ID.

**Word 8**

Is the address of an 82-character area into which the exit can place any error message text if the change request is denied or failed. Any text placed in this area is displayed to the user in response to the change request. If an error message is returned in this area it must be formatted as follows:

**Bytes 00 and 01**

Specifies the hexadecimal length of the message text (excluding these two bytes).

**Bytes 02 through *nn***

Specifies the error message text.

**Limits:** 80 characters

**Words 9 and 10**

Are set to zero.

## Return Codes from Change Call

The exit must return a completion code in Register 15 on return. The supported completion codes are described below. Specific causes for change rejection are identified by error message text returned in the area addressed by word 8 of the Change Call parameter list.

The return codes listed below are the only codes that are accepted in response to a Change Call. Any other return code is regarded as indicating that the password was not changed.

**0**

Indicates that the password has been changed.

**4**

Indicates that subfunction is not supported by exit or other error. A message explaining the cause of error might be placed in the 80-character area addressed by word 8 of the parameter list.

**8**

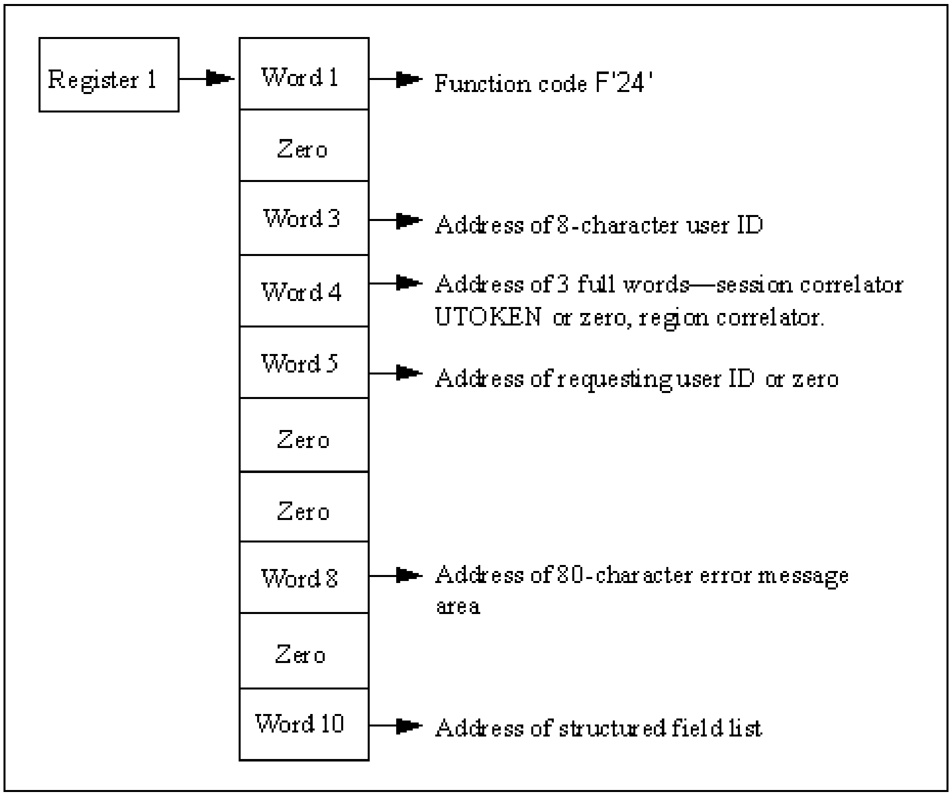
Indicates that the password has not been changed. A message explaining the cause of rejection is available in the 80-character area addressed by word 8 of the parameter list.

**Notes:**

- Your product region does not apply minimum password length checks, nor is automatic password expiry provided. The exit is responsible for all maintenance and knowledge of passwords.
- A user's current password is provided as a parameter for the change call only for subfunctions 0, 4, and 12. That is, when the change request is made by the user after logging on, when the user is required to change the password because it has expired or by EASINET using the &SECCALL CHECK statement.
- For subfunction 8, when a UAMS privileged user requests a forced password change for another user ID it is almost certainly because the current password for that user ID has been forgotten, and is therefore not available for presentation as a parameter for this call.

# Return User ID Information Parameter List

On entry to the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Function code F'24'
Zero
Word 3
Address of 8-character user ID
Word 4
Address of 3 full words—session correlator UTOKEN or zero, region correlator.
Word 5
Address of requesting user ID or zero
Zero
Zero
Word 8
Address of 80-character error message area
Zero
Word 10
Address of structured field list

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘24’.

**Word 2**

Is set to zero.

**Word 3**

Is the address of an eight-character field, blank-padded, containing the user ID for which information is required.

**Word 4**

Is the address of three full words containing the session correlator and the user token (UTOKEN) associated with the user issuing the request (not necessarily the correlator for the user ID whose information is being requested).

**Word 5**

Is the address of an eight-character field, blank-padded, containing the user ID requesting the information (or zero).

**Word 6**

Is set to zero.

**Word 7**

Is set to zero.

**Word 8**

Is the address of an 82-character area into which the exit can place any error message text if the function request is denied or failed. Any text placed in this area is displayed to the user in response to the command that caused this request. If an error message is returned it must be formatted as follows:

**Bytes 00 and 01**

Specifies the hexadecimal length of the message text (excluding these two bytes).

**Bytes 02 through *nn***

Specifies the error message text.

**Limits:** 80 characters

**Word 9**

Is set to zero.

**Word 10**

Is set differently depending on the type of exit you have:

- Full security exit—if the request is honored, the exit must place in this word the address of a list of full words, terminated by X‘FFFFFFFF’. Each fullword in this list in turn points to a structured field that defines an item of information relevant to the user ID.
- Partial security exit—the address of a list of full words, terminated by X‘FFFFFFFF’. Each fullword in this list in turn points to a structured field that defines an item of information relevant to the user ID.

**More information:**

Structured Fields (see page 79)

## Return Codes from Return User ID Information Call

The exit must return a completion code in Register 15 on return. Completion codes are supported as described below. Causes for rejection are identified by error message text returned in the area addressed by word 8 of the Information Call parameter list.

The return codes listed below are the only codes that are accepted in response to an Information Call. Any other return code is treated as information not available.

**0**

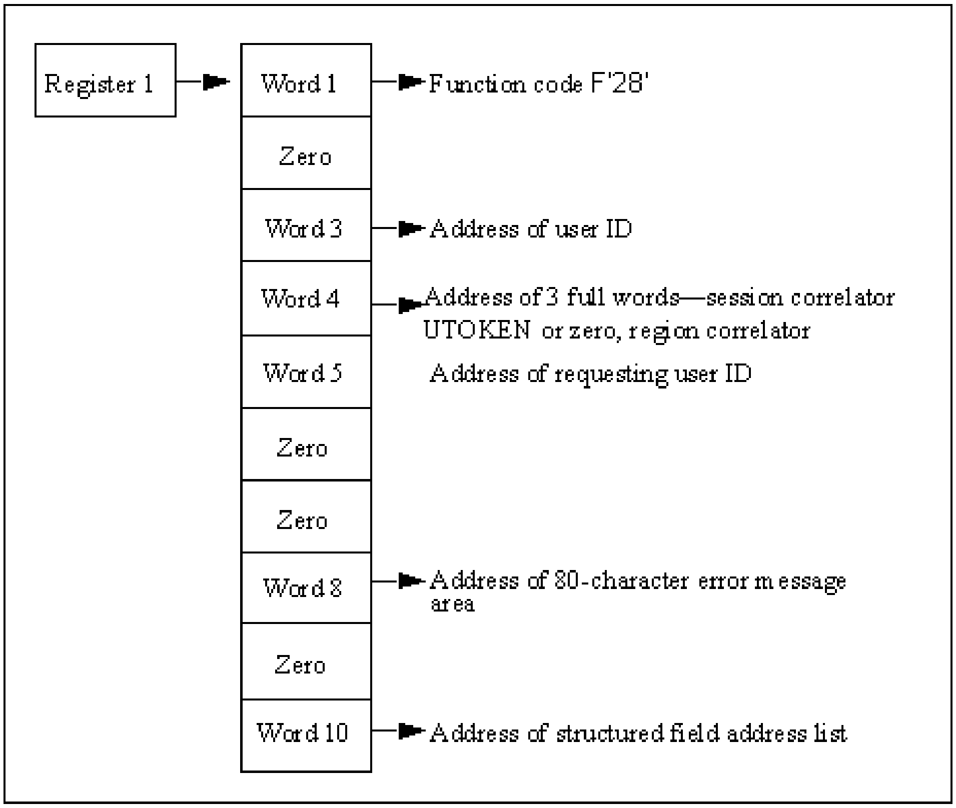
Indicates that information is available. Word 10 of the parameter list points to an address list that provides pointers to structured fields describing the user ID.

**4**

Indicates that information is not available. An error message should be available in the 80-character area addressed by word 8 of the parameter list.

## Update User ID Parameter List

On entry to the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Function code F'28'
Zero
Word 3
Address of user ID
Word 4
Address of 3 full words—session correlator
UTOKEN or zero, region correlator
Word 5
Address of requesting user ID
Zero
Zero
Word 8
Address of 80-character error message area
Zero
Word 10
Address of structured field address list

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘28’.

**Word 2**

Is set to zero.

**Word 3**

Is the address of an eight-character field, blank-padded, containing the name of the user ID to update.

**Word 4**

Is the address of three full words containing the session correlator and the user token (UTOKEN) associated with the user issuing the request.

**Word 5**

Is the address of an eight-character field containing the user ID of the user making the update request.

**Word 6**

Is set to zero.

**Word 7**

Is set to zero.

**Word 8**

Is the address of an 82-character area into which the exit can place any error message text if the function request is denied or failed. Any text placed in this area is displayed to the user in response to the command that caused this request. If an error message is returned in this area it must be formatted as follows:

**Bytes 00 and 01**

Specifies the hexadecimal length of the message text (excluding these two bytes).

**Bytes 02 through *nn***

Specifies the error message text.

**Limits:** 80 characters

Message text is converted to uppercase.

**Word 9**

Is set to zero.

**Word 10**

Is set to the address of a list of full words, terminated by X‘FFFFFFFF’. Each fullword in this list in turn points to a structured field that defines an item of user ID information to change.

**More information:**

Structured Fields (see page 79)

## Return Codes from Update User ID Information Call

The exit must return a completion code in Register 15 on return. Completion codes are supported as described below. Causes for update rejection are identified by error message text returned in the area addressed by word 8 of the Update Call parameter list.

The return codes listed below are the only codes that are accepted in response to an Update Call. Any other return code is regarded as update rejected.

**0**

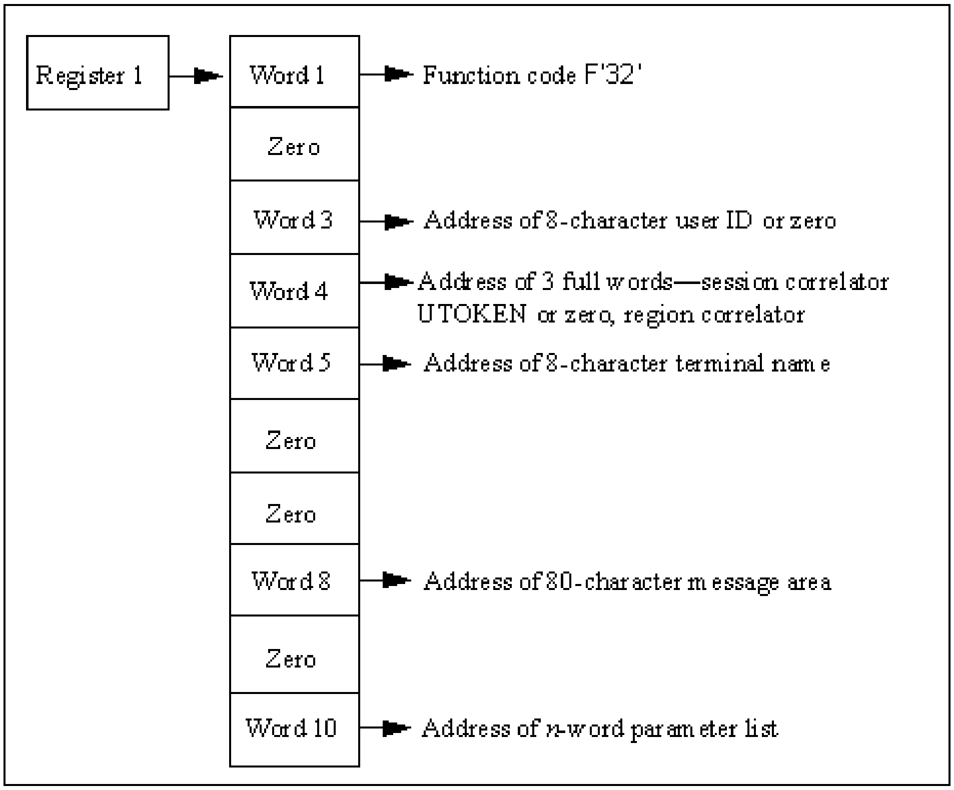
Indicates that update has completed.

**4**

Indicates that update is rejected. An error message might be available in the 80-character area addressed by word 8 of the parameter list.

# &SECCALL EXIT Parameter List

On entry to the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Function code F'32'
Zero
Word 3
Address of 8-character user ID or zero
Word 4
Address of 3 full words—session correlator UTOKEN or zero, region correlator
Word 5
Address of 8-character terminal name
Zero
Zero
Word 8
Address of 80-character message area
Zero
Word 10
Address of *n*-word parameter list

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘32’.

**Word 2**

Is set to zero.

**Word 3**

Is the address of an eight-character field, blank-padded, containing the user ID executing the procedure in which &SECCALL EXIT was issued. If &SECCALL EXIT was issued from an EASINET procedure, this word is set to zero.

**Word 4**

Is the address of three full words containing the session correlator and the user token (UTOKEN) associated with the user issuing the &SECCALL EXIT statement. If no user ID is involved because the &SECCALL EXIT statement has been issued from an EASINET procedure, this word is set to zero.

**Word 5**

Is the address of eight-character terminal name.

**Word 6**

Is set to zero.

**Word 7**

Is set to zero.

**Word 8**

Is the address of an 82-character area into which the exit can place any message text. If a message is placed in this area it is returned to the issuing NCL procedure in the system variable &SYSMSG. If a message is returned in this area it must be formatted as follows:

**Bytes 00 and 01**

Specifies the hexadecimal length of the message text (excluding these two bytes).

**Bytes 02 through *nn***

Specifies the error message text. Message text is converted to uppercase.

**Limits:** 80 characters

**Word 9**

Is set to zero.

**Word 10**

Is the address of a variable length list of contiguous full words, ended by a fullword containing a value of X‘FFFFFFFF’. Each word in the list points at an area that represents an NCL variable, where the format of this area is always:

**Bytes 00 and 01**

Specifies the length of parameter data present (excluding these two bytes).

**Bytes 02 through 257**

Specifies the parameter data, padded to 256 bytes with blanks.

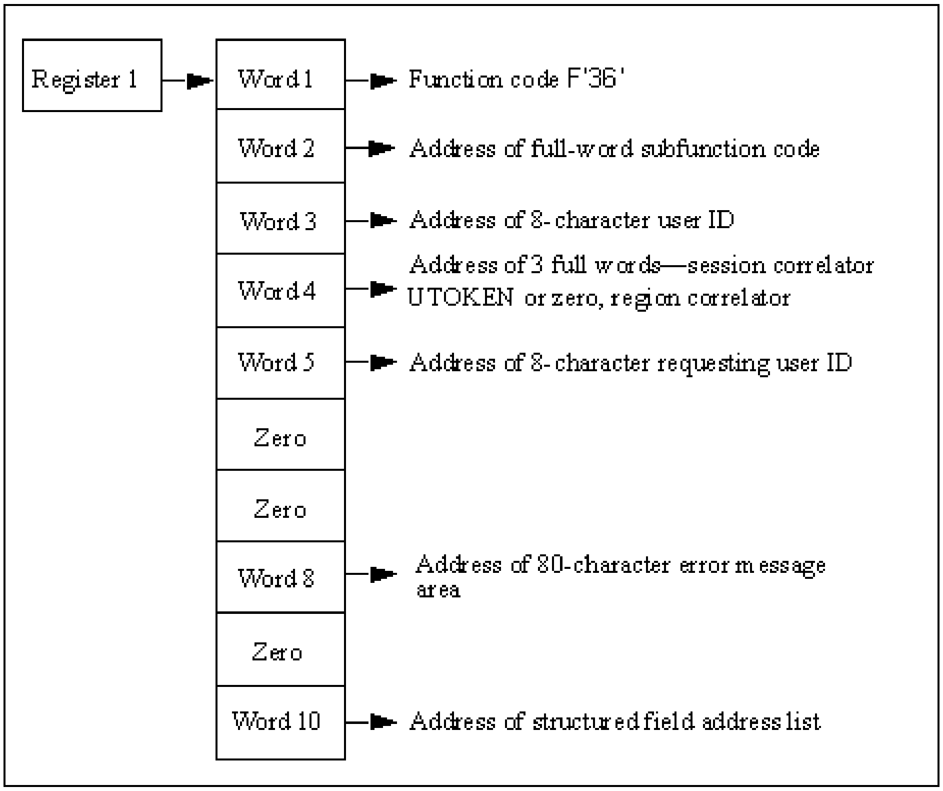
## Return Codes from &SECCALL EXIT Call

The exit can pass information back to the calling NCL procedure in the following ways:

- A return code can be set by the exit in Register 15 on exit. Valid return codes are 0 to 99. Any other return code is forced to 99. The return code is available to the procedure in the system variable &RETCODE after the &SECCALL EXIT statement.
- A message (error or otherwise) can be returned in the 82-character message area addressed by word 8 of the parameter list, as described above.
- Information can be returned in the parameter areas addressed by word 10 of the &SECCALL EXIT parameter list. The exit can return information only in those areas originally passed in the parameter list.

# Return Sequential User ID Information Parameter List

On entry to the security exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Function code F'36'
Word 2
Address of full-word subfunction code
Word 3
Address of 8-character user ID
Word 4
Address of 3 full words—session correlator UTOKEN or zero, region correlator
Word 5
Address of 8-character requesting user ID
Zero
Zero
Word 8
Address of 80-character error message area
Zero
Word 10
Address of structured field address list

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘36’.

**Word 2**

Is the address of a subfunction code. The code defines the type of sequential call being requested and has values set as follows:

| Subfunction Code | Value |
|---|---|
| F‘0’ | Retrieve the next user definition following the key user ID (provided in word 3). The user ID for which information is being requested is the next user ID in the collating sequence. |
| F‘4’ | Retrieve the previous user definition preceding the key user ID (provided in word 3). The user ID for which information is being requested is the previous user ID in the collating sequence. |

**Word 3**

Is the address of an eight-character field, blank-padded. The field contains the key user ID for the call, or set to zero if the request is issued for the first (next), or last (previous) user ID in the collating sequence.

**Word 4**

Is the address of three full words containing the session correlator and the user token associated with the user issuing the request (not necessarily the correlator for the user ID whose information is being requested).

**Word 5**

Is the address of an eight-character field containing the requesting user ID.

**Word 6**

Is set to zero.

**Word 7**

Is set to zero.

**Word 8**

Is the address of an 82-character area into which the exit can place any error message text if the function request is denied or failed. Any text placed in this area is returned to the issuing procedure in the &SYSMSG system variable. If an error message is returned it must be formatted as follows:

**Bytes 00 and 01**

Specifies the hexadecimal length of the message text (excluding these two bytes).

**Bytes 02 through *nn***

Specifies the error message text.

**Limits:** 80 characters

Message text is converted to uppercase.

**Word 9**

Is set to zero.

**Word 10**

If the request is honored, the exit must place in this word the address of a list of full words, terminated by X‘FFFFFFFF’. Each fullword in this list in turn points to a structured field that defines an item of information relevant to the user ID.

**More information:**

Structured Fields (see page 79)

## NWM--Return Codes from Return Next User ID Information Call

The exit must return a completion code in Register 15 on return. Completion codes are supported as described below. Causes for rejection are identified by error message text returned in the area addressed by word 8 of the parameter list.

The return codes listed below are the only codes that are accepted in response to the call. Any other return code is treated as return code 4.

**0**

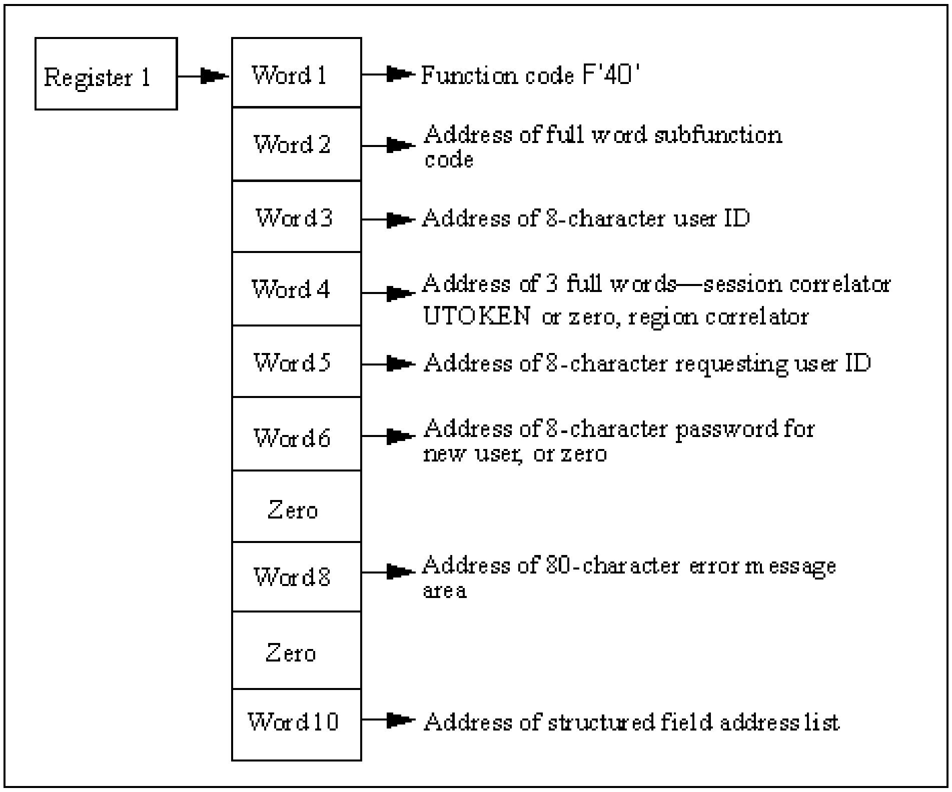
Indicates that information is available. Word 10 of the parameter list points to an address list that provides pointers to structured fields describing the user ID.

**4**

Indicates that information is not available. An error message might be available in the 80-character area addressed by word 8 of the parameter list.

# Add User ID Parameter List

On entry to the security exit, Register 1 (R1) points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Function code F'40'
Word 2
Address of full word subfunction code
Word 3
Address of 8-character user ID
Word 4
Address of 3 full words—session correlator UTOKEN or zero, region correlator
Word 5
Address of 8-character requesting user ID
Word 6
Address of 8-character password for new user, or zero
Zero
Word 8
Address of 80-character error message area
Zero
Word 10
Address of structured field address list

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘40’.

**Word 2**

Is the address of a subfunction code. The code defines the type of user to add and has values set as follows:

| **Subfunction Code** | **Value** |
|---|---|
| F‘0’ | Add a USER definition. |
| F‘4’ | Add a GROUP definition. |

**Word 3**

Is the address of an eight-character field, padded with blanks, containing the user definition to add.

**Word 4**

Is the address of three full words containing the session correlator and the user token (UTOKEN) associated with the user issuing the request.

**Word 5**

Is the address of an eight-character field containing the user ID of the user issuing the ADD request.

**Word 6**

Is the address of an eight-character field containing the initial password of the user to add or set to zero if a password is not provided.

This field is optional on the &SECCALL ADD function when using a security exit. The site is responsible to take appropriate action on the requirement for this password.

**Word 7**

Is set to zero.

**Word 8**

Is the address of an 82-character message area into which the exit can place any error message text if the add request is denied or failed. Any text placed in this area is displayed to the user in response to the command that caused this request. If an error message is returned in this area, it must be formatted as follows:

**Bytes 00 and 01**

Specifies the hexadecimal length of the message text (excluding these two bytes).

**Bytes 02 through *nn***

Specifies the error message text.

**Limits:** 80 characters

Message text is converted to uppercase.

**Word 9**

Is set to zero.

**Word 10**

Is set to the address of a list of full words, terminated by X'FFFFFFFF'. Each fullword in this list in turn points to a structured field that defines an item of user ID information to add.

**More information:**

Structured Fields (see page 79)

## Return Codes from the Add User ID Call

The exit must return a completion code in Register 15 (R15) on return. Completion codes are supported as described below. Causes for rejection of add calls are identified by error message text returned in the area addressed by word 8 of the add call parameter list.

The return codes listed below are the only codes that are accepted in response to an add call. Any other return codes are regarded as a rejection of the add call.

**0**

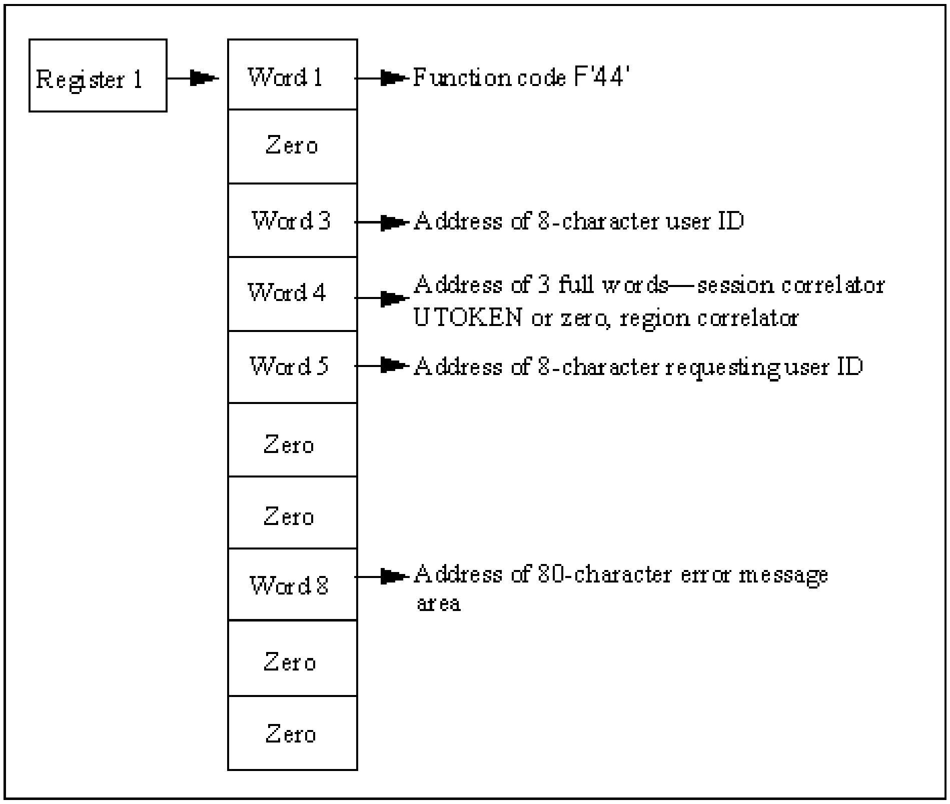
Indicates that ADD is complete.

**4**

Indicates that ADD is rejected. An error message might be available in the 80-character area addressed by word 8 of the parameter list.

# Delete User ID Parameter List

On entry to the security exit, Register 1 (R1) points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Function code F'44'
Zero
Word 3
Address of 8-character user ID
Word 4
Address of 3 full words—session correlator UTOKEN or zero, region correlator
Word 5
Address of 8-character requesting user ID
Zero
Zero
Word 8
Address of 80-character error message area
Zero
Zero

The contents of the parameter list are as follows:

**Word 1**

Is a fullword containing the address of function code F‘44’.

**Word 2**

Is set to zero.

**Word 3**

Is the address of an eight-character field, blank-padded, containing the user definition to delete.

**Word 4**

Is the address of three full words containing the session correlator and the user token (UTOKEN) associated with the user issuing the DELETE request.

**Word 5**

Is the address of an eight-character field containing the user ID of the user issuing the DELETE request.

**Word 6**

Is set to zero.

**Word 7**

Is set to zero.

**Word 8**

Is the address of an 82-character message area into which the exit can place any error message text if the delete request is denied or failed. Any text placed in this area is displayed to the user in response to the command that caused this request. If an error message is returned in this area, it must be formatted as follows:

**Bytes 00 and 01**

Specifies the hexadecimal length of the message text (excluding these two bytes).

**Bytes 02 through *nn***

Specifies the error message text.

**Limits:** 80 characters

Message text is converted to uppercase.

**Word 9**

Is set to zero.

**Word 10**

Is set to zero.

## Return Codes from the Delete User ID Call

The exit must return a completion code in Register 15 (R15) on return. Completion codes are supported as described below. Causes for delete rejection are identified by error message text returned in the area addressed by word 8 of the DELETE call parameter list.

The following return codes are the only codes that are accepted in response to a delete call. Any other return codes are regarded as a rejection of the delete call.

**0**

Indicates that Delete is complete.

**4**

Indicates that Delete is rejected. An error message might be available in the 80-character area addressed by word 8 of the parameter list.

# Appendix D: Data Set Authorization Exits Support

This section contains the following topics:

## Data Set Access Authorization Exit

The authorization exit is attached and operates as a subtask. Therefore, complex processing, WAITs, or SVCs do not impact overall performance.

Review the source code for the sample data set access authorization exit NMDSNCHK and use it as a guide to writing your own exit.

### Registers on Entry to the Exit

When the exit is invoked, Register 1 contains the address of a communication area containing various parameters. This communication area can be mapped using the macro $NMDSNCK, supplied in the distribution libraries. This macro provides a DSECT expansion to perform the mapping, and detailed information on the content of each field.

Standard linkage conventions apply. On entry, the exit must save the contents of all registers (Register 13 contains the address of a save area), and on exit all registers must be restored to their content on entry, with the exception of Register 15 which should contain a return code.

## Parameters Passed to the Exit

The data set access authorization exit is provided so that the installation can determine whether access to a data set is to be permitted.

- When invoked as a result of an ALLOCATE command, a communication area containing the following information is passed to the exit:
  - Data set name
  - User ID of requestor
- When invoked as a result of a CA SOLVE:FTS operation, a communication area containing the following information is passed to the exit:
  - Data set names
  - DD statement names
  - Class of transmission
  - Type of access required (READ at the sending end, READ/WRITE at the receiving end)
  - Whether this request is a SYSTEM or PRIVATE request
  - Type of data set allocation (NEW, SHR, and so on)
  - CA SOLVE:FTS ID
  - Transmission Definition password
  - CA SOLVE:FTS user ID of requestor
  - Space allocation and volume information, if this is a NEW data set

The format of this area is mapped by the supplied $NMDSNCK DSECT.

## Calls Made to the Exit

Whenever you use the ALLOCATE command to allocate a data set, a call is made to the authorization exit. A call is also made to the authorization exit when CA SOLVE:FTS identifies the data set that is to be transmitted or received. Calls are therefore made both at the transmitting and at the receiving end of a transmission operation.

On return from the initial call, the exit may set indicator flags (see the DSNCFLG field in the $NMDSNCK DSECT) that determine which subsequent calls CA SOLVE:FTS is to make to the exit during the progress of the transmission operation.

The following additional calls are made as requested by the exit:

- **Preallocation call**—made before the target data set is dynamically allocated
- **Pre-open call**—made just before the target data set is opened
- **Pre-deallocation call**—made just before the target data set is dynamically deallocated

### Types of Call

The DSNSTYPE field in the exit communication area identifies the type of call being made to the exit:

**F’0’**

Indicates a call for CA SOLVE:FTS access authorization.

**F’4’**

Indicates a call for the CA SOLVE:FTS allocation subtask.

**F’8’**

Indicates a call for the CA SOLVE:FTS open subtask or file.

**F’12’**

Indicates a call for the CA SOLVE:FTS deallocation subtask.

**F’16’**

Indicates a call for an ALLOCATE command.

If your site uses RACF security software, the exit can call RACF to associate the allocation with the requesting user ID rather than with your product region.

## Return Codes From the Exit

A return code is set in Register 15 on return from all calls to the exit, indicating the action that should be taken:

**0**

Indicates that access is permitted. Allocation can proceed.

**4**

Indicates that access is denied. An error message can be placed in the field DSNCTMSG, and its length in the field DSNCLMSG, in the communication area.

If the authorization exit is called by CA SOLVE:FTS and elects to receive the calls subsequent to the initial authorization call, return code 4, set in register 15 on return from any of those calls, will cause termination of the transmission at that point. The exit indicates which, if any, of the subsequent calls are required.

The additional calls to the exit allow greater control over the significant data set-specific operations that are involved in a transmission, letting the exit perform ENQ/DEQ functions to prevent duplicate access to data sets.

For RACF security software, the exit may call RACF at these times to associate the security responsibility for the action against the requesting user ID, rather than against CA SOLVE:FTS itself.

# Install the Data Set Access Authorization Exit

When you have created your own data set access authorization load module, place it in your product’s load library, or in another library concatenated to the load library through the STEPLIB DD statement in the started task JCL.

**To identify your exit to your product region**

1. Enter **/PARMS** at the command prompt. The Customizer : Parameter Groups list is displayed.
2. Enter **U** beside the NMSECURITY parameter group. The Customizer : Parameter Group panel for the NMSECURITY parameter group is displayed.
3. Enter the name of your load module in the Dataset Access Authorization Exit (NMDSNCHK) field.
4. Press F6 (Action) to set the changes; press F3 (File) to save the changes and exit.

# Data Set Services Authorization Exit

The data set services authorization exit is loaded by a data set services subtask and is called synchronously by the subtask. The exit can issue WAIT or other SVCs without impacting your product region's main task.

Review the source code for the sample data set services authorization exit NMDSSCHK and use it as a guide to writing your own exit.

The exit load module must be link edited as AMODE 24, RMODE 24. The exit procedure can be coded as reentrant; however, this is not required because an explicit load is performed for each task using the exit. The exit is always called in 24-bit address mode.

## Function Calls Made to the Exit

There are three main types of calls made to the exit:

- **Exit Initialization**—performs any initialization required.
- **Logical Function**—represents a function that has been requested by the user. For example, a data set allocation request or an OPEN request for a data set.
- **Exit Termination**—allows the exit to clean up any allocated storage areas and, if applicable, break the connection with any security subsystem that is being used.

Check the DSSCFUNC field to determine the type of call being made.

## Exit Environment

The exit is loaded separately by each invocation of the Data Services subtask. For example, a subtask is created to allocate or free a data set, or to open a data set, and so on.

The exit receives (per subtask invocation) one initialization call, possibly one or more logical function calls, and possibly one termination call. For example, copying a data set from A to B would result in the logical calls shown below:

```
ALLOC Atask init
ALLOC AALLOC call
ALLOC Atask terminate
ALLOC Btask init
ALLOC BALLOC call
ALLOC Btask terminate
READ A  task init
READ A  open check
WRITE Btask init
WRITE Bopen check
READ A  task terminate
WRITE Btask terminate
FREE A  task init
FREE A  ALLOC call
FREE A  task terminate
FREE B  task init
FREE B  ALLOC call
FREE B  task terminate
```

If the exit is defined as reentrant you can use the DSSCXCOR field to anchor any private storage required by the subtask.

**Note:** Do not use shared (SP 50) storage as an exit work area.

**More information:**

Exit Termination Call (see page 196)

## Registers on Entry to the Exit

On entry to the exit, Register 1 contains the address of a parameter list area that is used to pass information to the exit. This communication area is mapped by the $NMDSSCK macro, that is distributed with your product. This macro provides a DSECT expansion to perform mapping and contains detailed information on the contents of each field.

Standard module linkage conventions apply; on entry, the exit must save the contents of all registers (Register 13 contains the address of a save area that can be used by the exit) and on exit all registers must be restored to the value they had on entry with the exception of Register 15 that should contain a return code relevant to the function call.

## Exit Initialization Call

This call allows the exit to perform any initialization required. This could include authorizing the subtask against the authority level of the user. This call is not specific to any request from the user: logical function calls identify the functions that have been requested by a user and you can perform more specific processing for these calls.

The parameter area passed to the exit contains the user ID of the user, any User Token or Security Exit Correlator associated with the user (drawn from the main security exit, if one is being used) and information about the region in which the exit is running (for example, the NMID).

The exit must return with Register 15 set to a return code that indicates the success or failure of exit initialization. A return code can be set to indicate that no more security processing is required for the function.

The following return codes are allowed for this call:

**0**

Indicates that exit initialization completed successfully.

**4 (see page 195)**

Indicates that exit initialization completed successfully. No more logical function calls are required for this user—note that the termination call is still performed, however.

**8**

Indicates that exit initialization failed.

If Register 15 is set to 8, you can use the DSSCLMSG and DSSCTMSG fields to return a message indicating the reason for the failure. See the $NMDSSCK macro for full details. If no message is returned, data set services sets a default message. The message returned by the exit is recorded on the activity log.

**More information:**

Return Code 4 (see page 195)

## Return Code 4

If you set return code 4 on initialization, this means that the host does not need to participate on any subsequent logical function authorization.

You can set the return code, for example, if you issue a RACINIT during initialization to register the subtask as relating to a specific user ID (that is, when you have a security exit installed). In this case, for example, an OPEN DATASET call would result in an S913 OPEN ABEND because RACF does the security checking.

**Note:** Data set services handles this situation correctly.

## Return Code 0

If you set return code 0, but still connect the subtask to the security system, you can do additional checking on some calls (for example, rename PDS member) that the security system does not specifically check.

## Logical Function Call

This call represents a function that has been requested by the user. For example, a data set allocation request or a request to open a data set. The DSSCTYPE field is used to indicate the function to be performed.

The parameter area passed to the exit contains all information passed on the exit initialization call as well as any specific information requested by the user for the function. For example, if the call is for an allocation request the data set name, disposition, volume name, and so on, might be available. The $NMDSSCK macro contains a full description of each logical function call and the parameters that are passed on each call.

The exit can return with Register 15 set to a return code that indicates whether the function is permitted.

The following return codes are allowed for this call:

**0**

Indicates that the function is authorized.

**4**

Indicates that the function is not authorized.

If Register 15 is set to 4, a message indicating the reason for the failure can be returned using the DSSCLMSG and DSSCTMSG fields. See the $NMDSSCK macro for full details. If no message is returned by the exit, data set services sets a default message. The message returned by the exit is recorded on the activity log.

## Exit Termination Call

This call allows the exit to clean up any allocated storage areas and, if applicable, break the connection with any security subsystem that is being used. The exit load module is deleted from storage on return from this call.

**Note:** This call may not occur. If, for example, an NCL process using data set services is flushed, the subtask(s) are force detached. You should be aware of this and allocate any private storage in a non-shared subpool (this can include subpool SP0 but not SP50), which results in the storage being automatically freed.

The parameter passed to the exit contains all information passed on the exit initialization call. Register 15 must contain 0 on return from this call.

# Install the Data Set Services Authorization Exit

When you have created your own data set services authorization load module, place it in your product's load library, or in another library concatenated to the load library through the STEPLIB DD statement in the started task JCL.

**To identify your exit to your product region**

1. Enter **/PARMS** at the command prompt. The Customizer : Parameter Groups list is displayed.
2. Enter **U** beside the NMSECURITY parameter group. The Customizer : Parameter Group panel for the NMSECURITY parameter group is displayed.
3. Enter the name of your load module in the Dataset Services Authorization Exit (NMDSSCHK) field.
4. Press F6 (Action) to set the changes; press F3 (File) to save the changes and exit.

# Appendix E: INMC Security Exit Support

This section contains the following topics:

## INMC Security Exit

You must write a primary and secondary exit for each system that is to be controlled by the INMC security exit when connected by an INMC link.

The exits must be assembled and linked to form an executable load module and must be placed in a load library accessible to your product region during execution.

When the exit is called, standard assembler language linkage conventions apply.

When invoked, the exit operates under your product region's main task, and therefore extensive processing, I/O operations, or WAITs issued within the exit can impact the overall performance of the system.

### Identify the Primary Exit

One INMC primary exit is available for any one region and must be identified.

To identify the primary exit, include the following command in the INIT initialization procedure or issue the command before a link is activated:

```
SYSPARMS INMCEX01=exitname
```

***exitname***

Is the load module of the primary exit.

When an INMC link is activated, INMC attempts to load a copy of the exit. If the load fails for any reason it is regarded as a security exposure and the link is disabled automatically. Write the exit to be reentrant if possible because a load is issued for each link that is activated.

## Identify the Secondary Exit

One INMC secondary exit is available for any one system and must be identified.

To identify the secondary exit, include the following command in the INIT initialization procedure or issue the command before a link is activated:

```
SYSPARMS INMCEX02=exitname
```

***exitname***

Is the load module of the secondary exit.

When an INMC link is activated, INMC attempts to load a copy of the exit. If the load fails for any reason the primary exit of the remote system is notified that no secondary exit exists in this system. In this case the remote primary exit decides whether to allow link activation.

## Change Exit Names Dynamically

The name of the load module that forms either the primary or secondary INMC exit can be changed dynamically at any time.

To change an exit name, reissue one of the following commands to identify the new name:

```
SYSPARMS INMCEX01=new_primary_exitname
```

```
SYSPARMS INMCEX02=new_secondary_exitname
```

To disable an exit at any time, issue one of the following commands:

```
SYSPARMS INMCEX01=NONE
```

```
SYSPARMS INMCEX02=NONE
```

## Registers on Entering INMC Exits

Both primary and secondary exits are called using conventional linkage. The registers on entry to either exit contain values as follows:

| **Register** | **Value** |
|---|---|
| 0 | Unpredictable |
| 1 | Address of parameter list |
| 2 | Unpredictable |
| 3 | Unpredictable |
| 4 | Unpredictable |
| 5 | Unpredictable |
| 6 | Unpredictable |
| 7 | Unpredictable |
| 8 | Unpredictable |
| 9 | Unpredictable |
| 10 | Unpredictable |
| 11 | Unpredictable |
| 12 | Unpredictable |
| 13 | Address of standard save area |
| 14 | Product region return address |
| 15 | Entry point of INMC primary or secondary exit |

The exit must save registers on entry and perform standard save area linkage. On return, registers must be restored and control returned to the address held in Register 14 on entry.

# Primary Exit

Your product region calls the exit with Register 1 containing the address of a parameter list which is always ten consecutive full words in length.

Depending upon the reason for the call, some of these words are set to binary zeros.

The first word of the parameter list is always the address of a fullword that contains a function code identifying the type of call being made.

The other parameters passed depend upon the value of this function code, and the exit must therefore determine the function code first to decide which parameters to expect.

The parameter list passed to the exit is also used as a parameter list returned from the exit. The exit can thus indicate the processing required and pass the appropriate information to your product region.

**Note:** On a call to the exit, addresses of various fields are included in the parameter list. Only these fields can be used to return information from the exit; the exit cannot pass parameters back to your product region in any other locations.

## Initialization Processing

Initialization processing notifies the primary exit of two things:

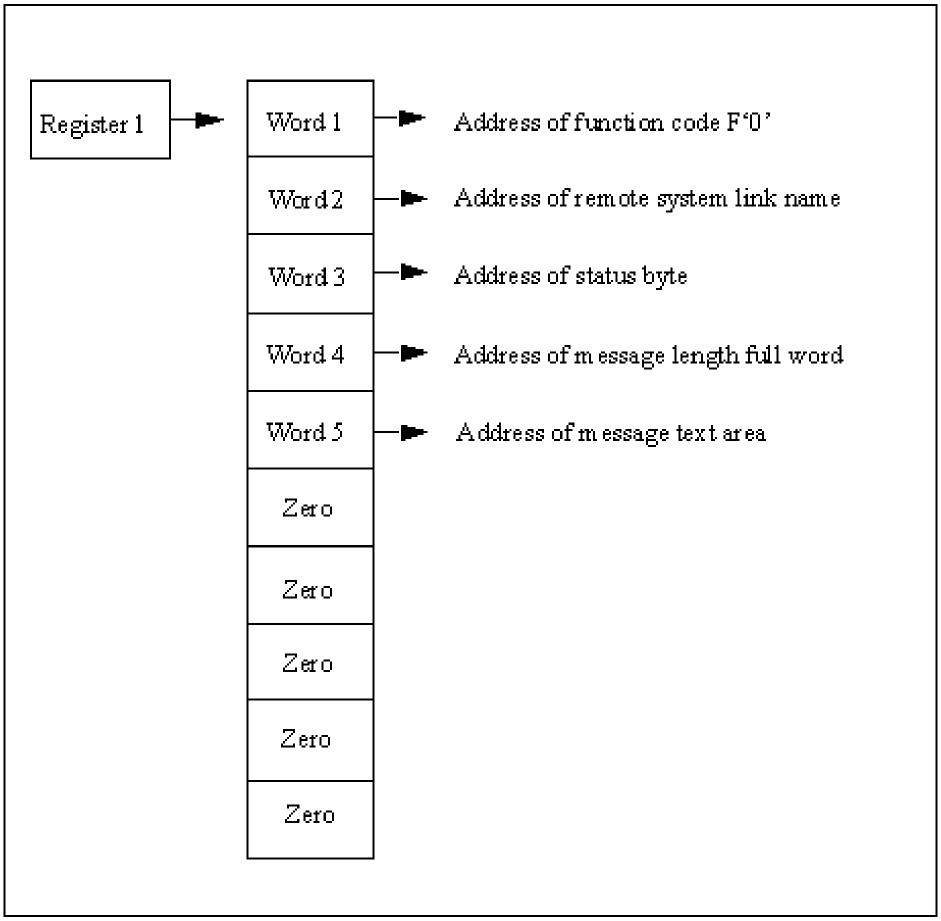
- The INMC link to the specified remote region activated
- Whether the remote region is configured with a secondary exit to which the primary exit can send messages

To provide this functionality in your exit, you code the initialization call to correspond with the parameter list in the following section.

Consider any special processing you want to perform, such as, password exchange between the connecting systems.

## Initialization Call Parameter List

On entry to the Primary exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'0'
Word 2
Address of remote system link name
Word 3
Address of status byte
Word 4
Address of message length full word
Word 5
Address of message text area
Zero
Zero
Zero
Zero
Zero

**Word 1**

Points to a full word containing function code X‘00000000’. Other words in the parameter list point to additional parameters.

**Word 2**

Is the address of the 1 to 12 character link name of the remote system with which a link has been established.

**Word 3**

Is the address of a 1-byte field that specifies the status of the secondary exit in the remote system. Values of this field and the meanings are as follows:

| Field Value | Definition |
|---|---|
| X‘00’ | A secondary exit is available for communications. |
| X‘04’ | The remote system does not have a secondary exit defined. No SYSPARMS INMCEX02 command has been issued in the remote system. |
| X‘08’ | The remote system has a secondary exit defined but the load module (or phase) was not loaded successfully. |

**Word 4**

Is the address of a full word, value F‘0’. The primary exit can place in this field the length of a message that is to be sent to the secondary exit in the remote system. The maximum length is 256 bytes (decimal). If a longer length is specified it is truncated to 256. If a negative length is specified it is forced to 256.

**Word 5**

Is the address of an area in which the primary exit can place the text of a message that is to be sent to the secondary exit in the remote system. The message is assumed to be left aligned in this message area. The maximum message text length is 256 (decimal) bytes. The data in the message can be binary or character and is transparent to your region. The meaning and format of the message sent to the secondary exit is determined by the installation.

**Words 6 to 10**

Are set to zero.

## Return Codes from Initialization Call

On return from the initialization call, the primary exit signals what it wants done next. The choices that are available are:

- Send a message to the secondary exit in the remote system.
- Terminate exit processing.

The exit indicates which option is required by using the same ten full words of the parameter list with which it was called, to pass back its own instructions to your region. When the exit returns therefore, the ten full word parameter list must be formatted as follows:

**Word 1**

Is the address of a function code specifying the action that the exit wishes your region to take. Values of this function code and their meanings are as follows:

| Function Code | Definition |
|---|---|
| F‘0’ | A message is to be sent to the secondary exit in the remote system. |
| F‘4’ | Exit processing is to terminate. |

**Word 2**

Is unchanged and not used.

**Word 3**

Is the address of a 1-byte field. This is the same address as was passed to the exit in word 3 of the parameter list. The value of this 1-byte field depends upon the function code returned by the exit in word 1. Valid values and their meanings are:

| Function Code | | Meaning |
|---|---|---|
| F‘0’ | | Function code not used. |
| F‘4’ | X‘00’ | Allow normal link activation |
| | X‘04’ | Allow link to remain open but allow no traffic to flow |
| | X‘08’ | Close the link |

**Word 4**

Is not used for function code F‘4’. For function code F‘0’ this word holds the address of a full word in which the primary exit can place the binary length of a message that is to be sent to the secondary exit in the remote system. The maximum length is 256 bytes (decimal). If a longer length is specified it is truncated to 256. If a negative length is specified it is forced to 256. This is the same address as passed to the exit in word 4 of the parameter list.

**Word 5**

Is not used for function code F‘4’. For function code F‘0’ this word holds the address of an area in which the primary exit can place the text of a message that is to be sent to the secondary exit in the remote system. The message is assumed to be left aligned in this message area. The maximum message text length is 256 (decimal) bytes. The data in the message can be binary or character and is transparent to your region. The meaning and format of the message sent to the secondary exit is determined by the installation. This is the same address as passed to the exit in word 5 of the parameter list.

**Words 6 to 10**

Are set to zero.

## Message Delivery Processing

When the primary exit sends a message to the secondary exit in a remote system, the secondary exit always responds with a reply. That reply can be a null message of zero length. The reply message is presented to the primary exit for processing by using a Deliver Call.

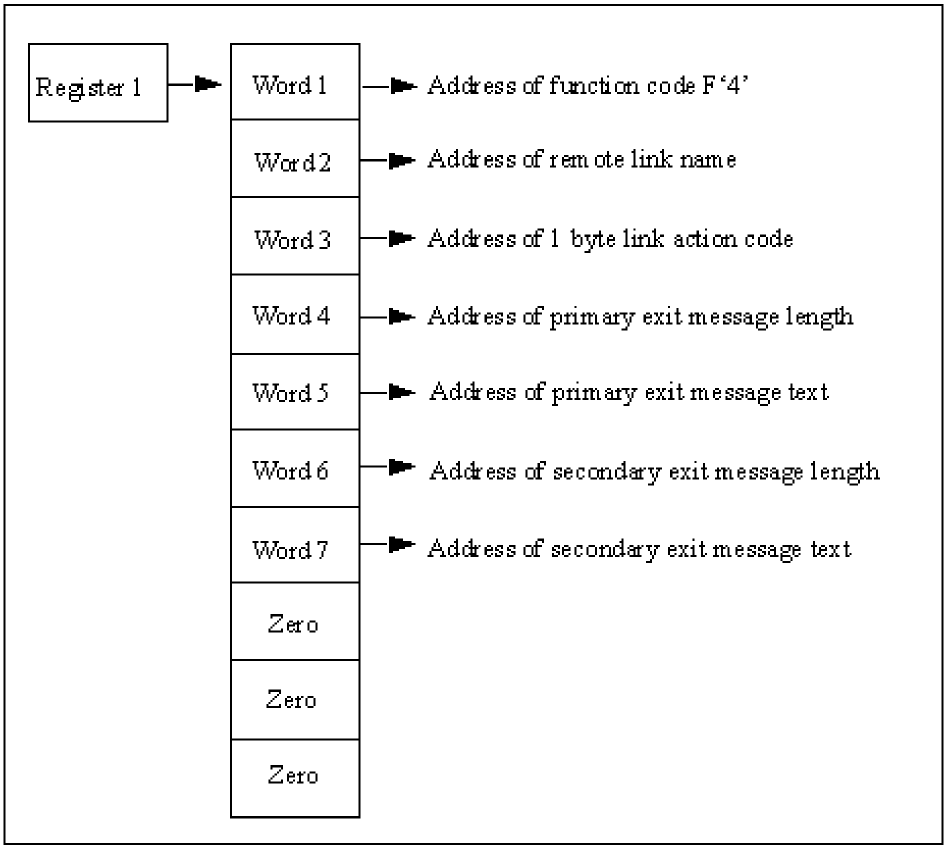
The primary exit performs whatever processing is required and then returns to your region with the ten fullword parameter list set to indicate the processing required next.

On return, the exit can request one of two options:

- Terminate exit processing.
- Send a message to the secondary exit.

## Delivery Call Parameter List

On entry to the primary exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'4'
Word 2
Address of remote link name
Word 3
Address of 1 byte link action code
Word 4
Address of primary exit message length
Word 5
Address of primary exit message text
Word 6
Address of secondary exit message length
Word 7
Address of secondary exit message text
Zero
Zero
Zero

**Word 1**

Points to a full word containing function code X'00000004'.

**Word 2**

Is the address of the link name of the remote system with which a link has been established.

**Word 3**

Is the address of a full word in which the primary exit can return a code indicating whether the link is to be opened for normal INMC traffic.

**Word 4**

Is the address of a full word, value F‘0’. The primary exit can place in this field the length of a message that is to be sent to the secondary exit in the remote system. The maximum length is 256 bytes (decimal). If a longer length is specified it is truncated to 256. If a negative length is specified it is forced to 256.

**Word 5**

Is the address of an area in which the primary exit can place the text of a message that is to be sent to the secondary exit in the remote system. The message is assumed to be left aligned in this message area. The maximum message text length is 256 bytes (decimal). The data in the message can be binary or character and is transparent to your region. The meaning and format of the message sent to the secondary exit is determined by the installation.

**Word 6**

Is the address of a full word containing the length of a message sent from the secondary exit in the remote system. The maximum length is 256 bytes (decimal), minimum is zero (which would be a null message).

**Word 7**

Is the address of the message sent from the secondary exit.

**Words 8 to 10**

Are set to zero.

## Return Codes from Delivery Call

When the primary exit completes its processing of the message returned from the secondary exit it formats the ten word parameter list to indicate the processing required next and then returns to your region.

The parameter list must be set exactly.

**More information:**

Return Codes from Initialization Call (see page 205)

## Termination of Link Notification Processing

If a link to a remote region is lost before a reply is received from the secondary exit, your region indicates this condition to the primary exit through a notify call. This call tells the primary exit that no further communications can be received from the secondary exit and that the INMC link to the remote region has been lost.

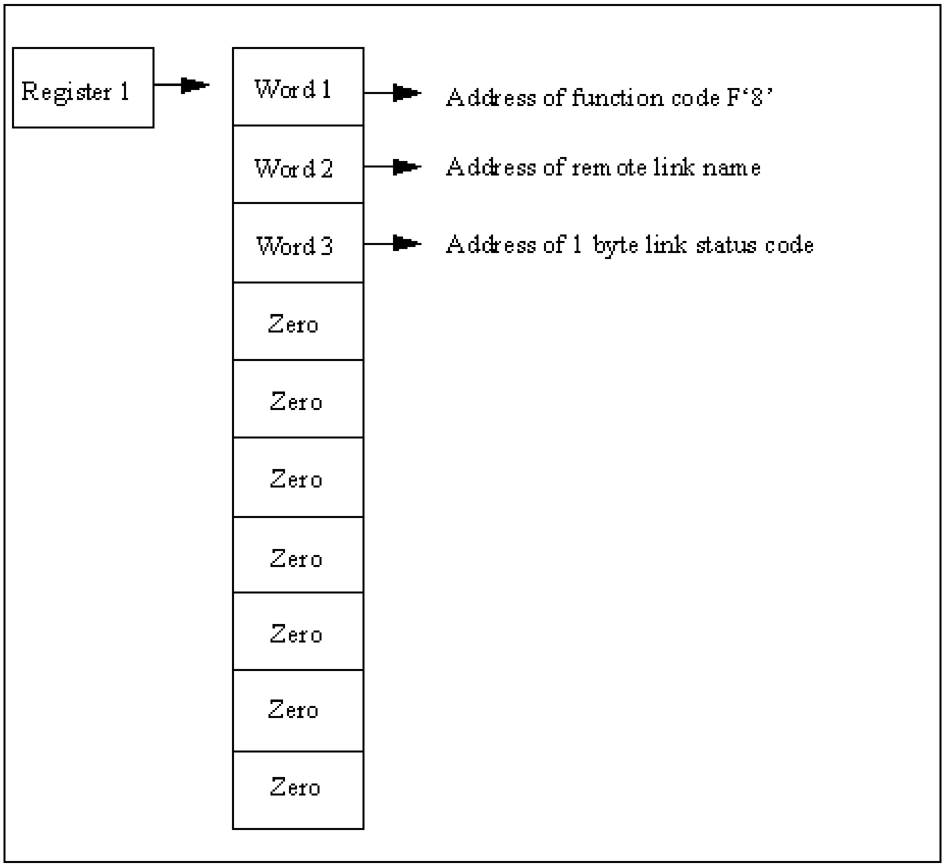
A notify call is also made if the primary exit attempts to communicate with a secondary exit under the following conditions:

- The remote region is not configured with a secondary exit.
- The secondary exit failed to initialize.

When the exit is notified of a lost link, it performs any required termination processing and then return to your region, indicating that exit processing is to terminate. Any other action results in repetitive notify calls until the exit signals that exit processing is to end.

## Notify Call Parameter List

On entry to the primary exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'8'
Word 2
Address of rem ote link name
Word 3
Address of 1 byte link status code
Zero
Zero
Zero
Zero
Zero
Zero
Zero

**Word 1**

Points to a full word containing function code X'00000008'.

**Word 2**

Is the address of the link name of the remote system with which contact has been lost.

**Word 3**

Is the address of 1-byte field containing a notify code. Valid values are as follows:

| Field Value | Definition |
|---|---|
| X‘04’ | The remote system does not have a secondary exit defined. No SYSPARMS INMCEX02 command has been issued in the remote system. |
| X‘08’ | The remote system has a secondary exit defined but the load module (or phase) was not loaded successfully. |
| X‘0C’ | The link has been lost. |

**Words 4 to 10**

Are set to zero.

## Return Codes from Notify Call

When the primary exit completes its processing of the notify call it should format the ten word parameter list as follows:

**Word 1**

Is the address of a function code specifying the action that the exit requires your region to take. The values of this function code must be F’4’ (Exit processing is to terminate).

The remaining nine full words should be returned unchanged.

# Secondary Exit

Your product region calls the exit with Register 1 containing the address of a parameter list which is always ten consecutive full words in length.

Depending upon the reason for the call, some of these words are set to binary zeros.

The first word of the parameter list is always the address of a fullword that contains a function code identifying the type of call being made.

The other parameters passed depend upon the value of this function code, and the exit must therefore determine the function code first to decide which parameters to expect.

The parameter list passed to the exit is also used as a parameter list returned from the exit. The exit can thus indicate the processing required and pass the appropriate information to your product region.

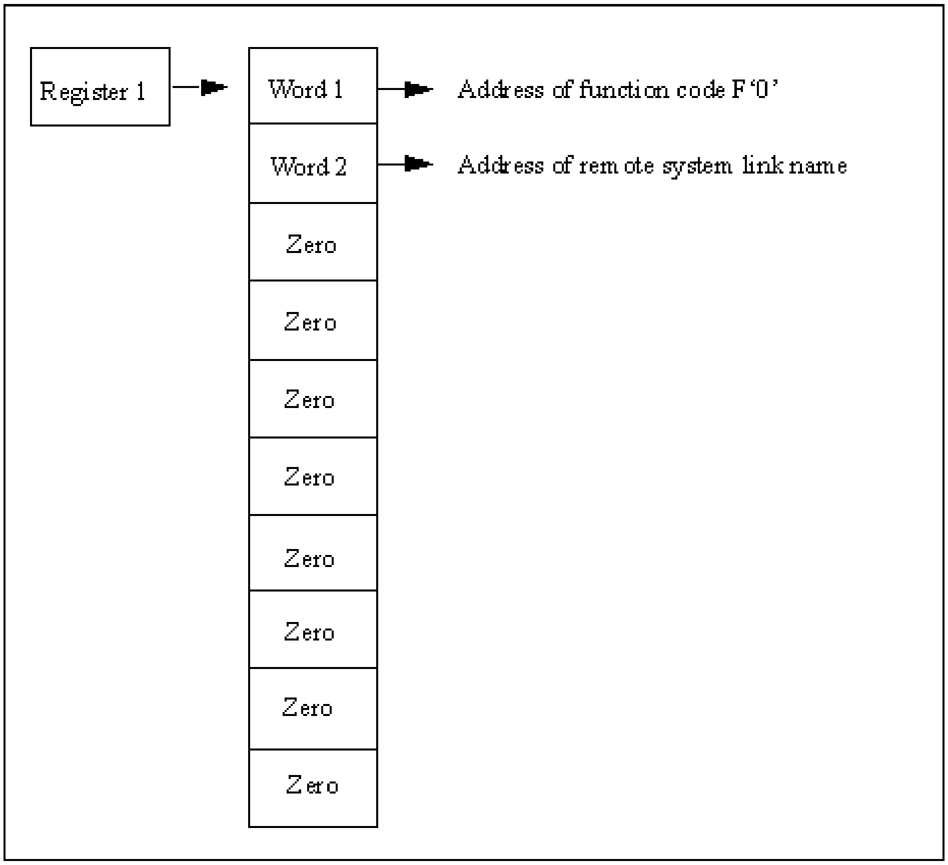
**Note:** On a call to the exit, addresses of various fields are included in the parameter list. Only these fields can be used to return information from the exit; the exit cannot pass parameters back to your product region in any other locations.

## Initialization Processing

Initialization processing indicates to the secondary exit that a link has become active. There is no indication whether a primary exit exists in the remote system, or, if there is, whether that exit is going to attempt to communicate with the secondary exit. The secondary exit must therefore be written to expect whatever processing is implemented in the remote primary exit.

## Initialization Call Parameter List

On entry to the secondary exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'0'
Word 2
Address of remote system link name
Zero
Zero
Zero
Zero
Zero
Zero
Zero
Zero

**Word 1**

Points to a full word containing function code X'00000000'. Other words in the parameter list point to additional parameters.

**Word 2**

Is the address of the link name of the remote system with which a link has been established. There may or may not be a primary exit defined for that system. If there is, it is up to the primary exit to determine whether it wants to communicate with this secondary exit.

**Words 3 to 10**

Are set to zero.

### Return Codes from the Initialize Call

The secondary exit does not have an opportunity to respond to the initialize call. On completion of any processing it chooses to perform when passed control, the secondary exit should return to your region with its ten full word parameter list unchanged.

## Message Delivery Processing

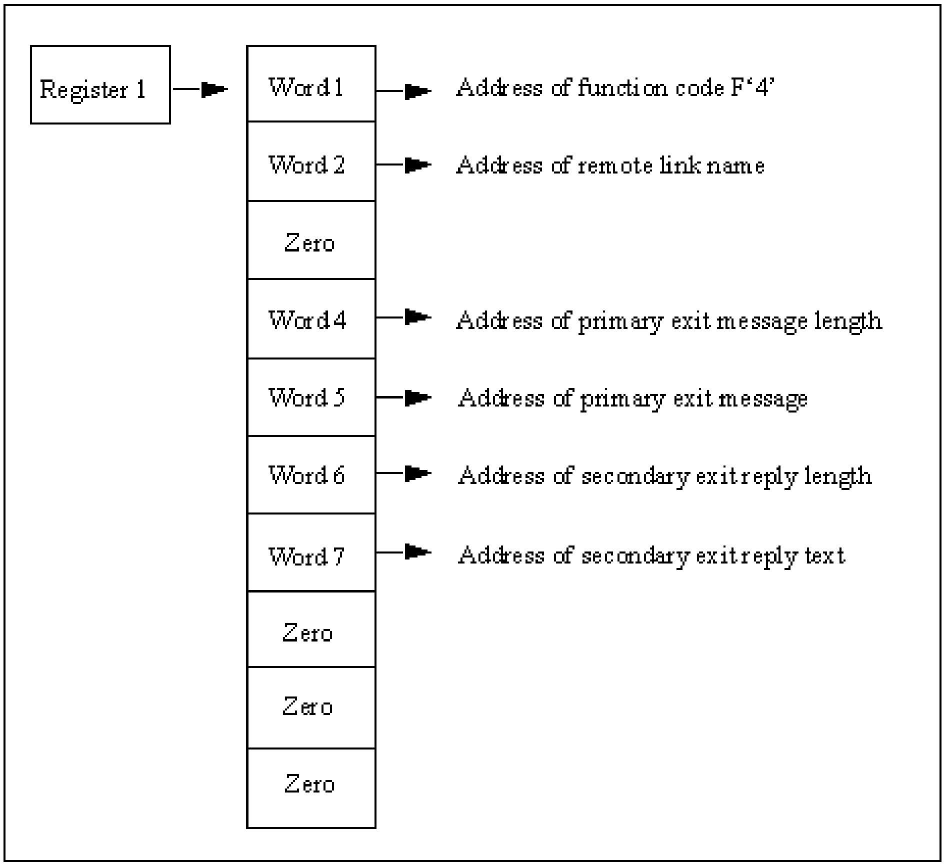
When the primary exit sends a message to the secondary exit the message is presented to the secondary exit for processing using the Deliver Call.

The secondary exit performs whatever processing is required and then returns to your region. On return, your region expects the secondary exit to have set the message-length and message-text areas pointed to by words 6 and 7 of the parameter list with which it was called.

The secondary exit can, if necessary, indicate that a null message is to be returned to the primary exit, perhaps as an acknowledgment to the message sent by the primary.

## Deliver Call Parameter List

On entry to the secondary exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F‘4’
Word 2
Address of remote link name
Zero
Word 4
Address of primary exit message length
Word 5
Address of primary exit message
Word 6
Address of secondary exit reply length
Word 7
Address of secondary exit reply text
Zero
Zero
Zero

**Word 1**

Points to a full word containing function code X‘00000004’.

**Word 2**

Is the address of the link name of the remote system from which the message has been received.

**Word 3**

Is set to zero.

**Word 4**

Is the address of a full word containing the length of a message sent from the primary exit in the remote system. The maximum length is 256 bytes (decimal), minimum is zero (which would be a null message).

**Word 5**

Is the address of the message sent from the primary exit.

**Word 6**

Is the address of a full word, value F'0'. The secondary exit can place in this field the length of a message that is to be sent to the primary exit in the remote system. The maximum length is 256 bytes (decimal). If a longer length is specified it is truncated to 256. If a negative length is specified it is forced to 256.

**Word 7**

Is the address of an area in which the secondary exit can place the text of a message that is to be sent to the primary exit in the remote system. The message is assumed to be left aligned in this message area. The maximum message text length is 256 bytes (decimal). The data in the message can be binary or character and is transparent to your region.

The meaning and format of the message sent to the primary exit is determined by the installation.

**Words 8 to 10**

Are set to zero.

## Return Codes from Secondary Exit Deliver Call

The secondary exit should set the full word pointed to by word 6 of the call parameter list to the length of the message text that is to be returned to the primary exit as a reply to the message just delivered.

The message length has a range of 0 to 256 in hexadecimal. Any text outside the specified range is truncated to 256 bytes decimal. A zero length is accepted. Negative length settings are forced to a zero value.

Word 7 of the call parameter list points to a 256-byte area in which the secondary exit can place the text of the message to be returned to the primary exit. The format and content of the message returned is decided by the installation.

No other return information is accepted from the secondary exit.

## Termination Processing

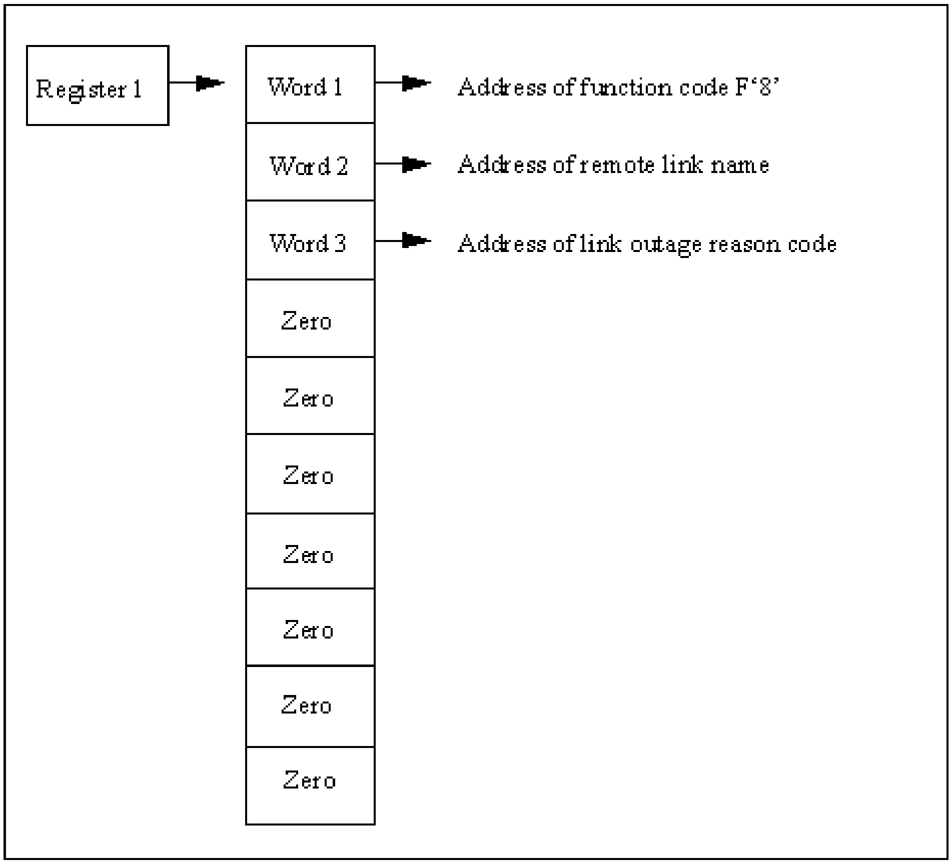
When the primary exit in a remote region decides that the exchange of information between the primary and secondary exits is complete, it returns to your region. The exit indicates that exit processing is to terminate, and specifies whether to activate or close down the link.

The decision to finish communication is reported to the secondary exit as a disconnect call. The secondary exit determines that no further messages are to be received from the primary exit and releases any exit-dependent resources it acquired.

The disconnect call is also issued when the link to the primary exit's remote system is lost.

### Disconnect Call Parameter List

On entry to the secondary exit, Register 1 points to a list of ten full words, as shown in this diagram.


Register 1
Word 1
Address of function code F'8'
Word 2
Address of remote link name
Word 3
Address of link outage reason code
Zero
Zero
Zero
Zero
Zero
Zero
Zero

**Word 1**

Points to a full word containing function code X‘00000008’.

**Word 2**

Is the address of the link name of the remote system from which the message has been received.

**Word 3**

The address of 1 byte disconnect reason code. Valid values are:

| Field Value | Definition |
|---|---|
| X‘00’ | Orderly termination of communication by choice of the primary exit. |
| X‘04’ | Disconnection caused by loss of contact. |

**Words 4 to 10**

Are set to zero.

## Return Codes from the Disconnect Call

The secondary exit should clean up any resources allocated during the conversation with the primary exit and perform any required termination processing. No further communication is possible with the primary exit.

No parameters can be returned to your region.

On return to management service, the link either is out of action (if caused by a link outage) or is activated or deactivated according to the choice of the primary exit in the remote system.

# Appendix F: NMSAF Public Correlator

This section contains the following topics:

## Overview

The NMSAF solution is built around the partial security exit facility.

Your product provides the security exit with the ability to supply the following information for each user that logs on:

- A security correlator—This correlator is typically used to anchor useful control blocks (such as an ACEE).
- A UTOKEN pointer—This pointer (if used) must point to a valid SAF UTOKEN that your product region can use.

Source for the NMSAF exit is not supplied. However, to allow you to write other exits that can take advantage of the information that it associates with a user, part of the security correlator information is made public.

For example, you can write an NCLEX01 exit that performs resource checking using the ACEE associated with a user.

The NMSAF solution also allows you to set various global options by using the USER*xxx* options in the SXCTL file. A facility is provided that allows you to locate this information.

**More information:**

SXCTL Parameters (see page 67)

# Security Correlator

When you are using NMSAF, any exit for your product region (such as NMDSNCHK and NMDSSCHK) that receives the security correlator can access the public information.

The security correlator includes:

- A storage area, mapped by the DSECT in the $NMUCORH macro

  This area contains data such as the product region user ID and the SAF user ID (which can be different), the ACEE address, and the user logon type.

- A global area, mapped by a provided DSECT

  This area contains the values of the global user flags and fields.

# $NMUCORH  Macro

With the NMSAF security facility, you can use the $NMUCORH macro to map the area pointed to by the security correlator field provided in many exits in your product region.

The first part of the area is regarded as public but is read-only. Each logged on user has a separate area (if a user ID is logged on more than once, each instance has a separate area).

**Important!** Do not alter any of the fields.

## DSECTs in the $NMUCORH Macro

The $NMUCORH macro expands into two DSECTs:

- A UCOR DSECT (if you place a label on the macro request, that label is used as the DSECT name instead of UCOR)
- A UGIN DSECT, which maps the global information area (see below)

## Fields in the UCOR DSECT

The UCOR (or renamed) DSECT contains the following fields. For more information about these fields, including equated values, see the source code of the $NMUCORH macro.

**Important!** This list shows only the public header portion of the area. Other fields follow the header. However, they are not documented and must not be altered.

**UCOREYEC**

Specifies an eye catcher.

**Type:** Character

**UCORUID**

Specifies the user ID.

**Type:** Character

**UCORSUID**

Specifies the SAF user ID, which is normally UCRUID, except when a system region and a common SAF user ID is used, or when system regions are not SAF validated (in which case this field is blank).

**Type:** Character

**UCORUTYP**

Specifies the user logon type. It can have various equated values.

**Type:** Number

**UCOR@ACE**

Specifies the address of the SAF ACEE for this user (0 if no SAF user ID).

**Type:** Address

**UCOR@TOK**

Specifies the address of the UTOKEN area (generally provided as a parameter to the exit as well).

**Type:** Address

**UCORUGIN**

Specifies the address of the UGIN global area. This is a constant value and points to the area mapped by the UGIN DSECT.

**Type:** Address

**UCOR@USX**

Specifies the address of a site-specific extension area. It is initially zero.

**Type:** Address

**UCORLUSX**

Specifies the length of a site-specific extension area. It is initially zero.

**Type:** Number

**UCORUSER**

Specifies anything that can be used by user exits.

**Limits:** Six full words (24 bytes)

**Type:** *

You can use the UCOR@USX and UCORLUSX fields to provide the address and length of a site-specific extension:

- If they are *both* set, then the NMSAF exit issues a FREEMAIN when the user correlator is freed (during logoff processing). The storage *must* be allocated in your product region shared subpool (SP 50).
- Setting only the address does not cause a FREEMAIN; this is handy for anchoring, for example, some global area.

**More information:**

Fields in the UGIN DSECT (see page 223)

## Fields in the UGIN DSECT

The UGIN DSECT contains the following fields. This is a common area that contains global information. Some of it is system information and some is available for your use.

**UGINEYEC**

Is an eye catcher.

**Type:** Character

**UGINSFLG**

Is the system flag area (eight flags).

**Type:** Bits

**UGINSFL1** to **UGINSFL8**

Are the system flags.

**Type:** Bits

**UGINSCFL**

Is the system character field area (8-byte fields).

**Type:** Character

**UGINSCF1** to **UGINSCF8**

Are the character fields.

**Type:** Character

**UGINUPT1** to **UGINUPT4**

Are the pointer fields (4-byte fields) that are available for installation use (any storage anchored must be obtained in SP 50).

**Type:** Pointer

**UGINUFL1** to **UGINUFL8**

Are eight user flags set by the USERFLAG*n* input file statements in SXCTL (corresponding to the 8 bits in the byte 80..01).

**Type:** Bits

**UGINUCF1** to **UGINUCF8**

Are the eight-character fields that can be set by SXCTL for input file statements by using NMSAF.

UGINUCF1 to UGINUCF4 are user names, as set by USERNAME*n*.

UGINUCF5 to UGINUCF8 are user IDs, as set by USERUID*n*.

# Appendix G: External Security Definitions for Modeled Users

This section contains the following topics:

## Define Resources in CA ACF2

To define and activate the resources in CA ACF2, issue the following commands in TSO:

```
[ACF]
SET RESOURCE(FAC)
COMPILE *
$KEY(SOLVE) TYPE(FAC)
ADMIN UID(USER1) SERVICE(READ) ALLOW
OPER  UID(USER2) SERVICE(READ) ALLOW
NOPER UID(USER3) SERVICE(READ) ALLOW
MON   UID(*)     SERVICE(READ) ALLOW
STORE
[END]
```

**Note:** Instead of using TSO, you can use the ACFBATCH utility in JCL. If you use the utility, omit the [ACF] and [END] lines.

## Define Resources in CA Top Secret

To define and activate the resources in CA Top Secret, issue the following commands in TSO:

```
TSS  ADD(dept)     IBMFAC(SOLVE)
TSS  PER(USER1)    IBMFAC(SOLVE.ADMIN)
TSS  PER(USER2)    IBMFAC(SOLVE.OPER)
TSS  PER(USER3)    IBMFAC(SOLVE.NOPER)
TSS  PER(USERPROF) IBMFAC(SOLVE.MON)
```

# Define Resources in RACF

**To define and activate the resources in RACF**

1. Issue the following RACF commands:

   ```
   RDEFINE FACILITY SOLVE.ADMIN UACC(NONE)
   RDEFINE FACILITY SOLVE.OPER  UACC(NONE)
   RDEFINE FACILITY SOLVE.NOPER UACC(NONE)
   RDEFINE FACILITY SOLVE.MON   UACC(READ) (see note)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

   **Note:** If you do not want to allow general browse access, specify:

   ```
   RDEFINE FACILITY SOLVE.MON   UACC(NONE)
   ```

2. To connect users to the resources, issue commands like the following:

   ```
   PERMIT SOLVE.ADMIN  CLASS(FACILITY)  ID(USER1)
   PERMIT SOLVE.NOPER  CLASS(FACILITY)  ID(USER2)
   ```

   **Note:** The default access is READ, which is sufficient.

# Appendix H: Command Authority Levels

Each product command has a default authority level that is required for execution.

This section contains the following topics:

## Overview

Each product command has an assigned authority level in the range 0 through 255. For a user to execute a particular command, their corresponding command authority level must be equal to or higher than the authority level assigned to the command. Command authority levels for users are allocated in their user ID definition.

Default command authority levels have been allocated to each product command according to the importance or power associated with each command. For example, VTAM display commands have an authority level of 0. A user with a command authority level of 0 can execute display commands. They cannot execute commands that affect or change the status of the network because those commands require a higher authority.

The default command authority level of product commands can be changed.

# Command Authority Summary Table

The following table describes the available commands and their default authority level. Where a special authority applies to certain operands of a command, the authority levels are expressed as normal/special, for example, 0/2. You can use the *opauth* sub-parameter of the SYSPARMS CMDAUTH operand on these commands.

**Note:** For more information about using this operand, see the *Reference Guide*.

Commands with an authority level of T are controlled by the TCP/IP access level.

**Note:** For more information about each product command, see the online help.

| Command | Description | Auth |
|---|---|---|
| ACT | Activates a VTAM network node | 1 |
| ACTLOGON | Activates previously suspended logon paths | 3 |
| AFTER | Issues commands or a message after the specified time interval | 0/2 |
| ALLOCATE | Allocates a cataloged, sysout, or VSAM data set | 0/3 |
| AOM GO | Restarts the local AOM operating system after a pause | 2 |
| AOM NEWTAB | Compiles and (optionally) loads a new AOM Screening Table | 2 |
| AOM PAUSE | Suspends the local AOM operating system interface | 2 |
| AOM START | Starts the local AOM operating system interface | 2 |
| AOM STOP | Terminates the local AOM operating system interface | 2 |
| AT | Issues commands or a message AT the specified time of day | 0/2 |
| ATF DEFINE | Defines a new translation name | 2 |
| ATF LOAD | Loads translation tables for a specific translation name into storage | 2 |

| Command | Description | Auth |
|---|---|---|
| ATF UNLOAD | Unloads translation tables for a specific translation name | 2 |
| AUTOHOLD | Determines the OCS screen display line mode | 0 |
| CANCEL | Cancels a nominated user or user region or window | 0/1 |
| CLEAR | Clears the current logical screen window | 0 |
| CLSDST | Closes the sessions with the specified terminals | 1/0 |
| CNM | Starts and stops the VTAM CNM interface | 3 |
| CNMTRACE | Defines the CNM records to trace | 3 |
| CONNECT | Connects a terminal using XNF | 1 |
| CS- | Retrieves the command from the bottom of the command stack | 0 |
| CS+ | Retrieves the command from the top of the command stack | 0 |
| D | Displays a specific VTAM resource | 0 |
| DEBUG | Controls the execution and debugging of NCL processes | 0/3 |
| DEFALIAS | Defines an alias entry for the Alias Name Translation Facility of NEWS | 4 |
| DEFCLASS | Defines RTM, SAW, Session, or Resource classes | 4 |
| DEFLINK | Maintains the Dynamic Link Table, or DLT, used by LU6.2 services | 3 |
| DEFLOGON | Defines the logon information for the EASINET and MAI features | 4 |
| DEFMODE | Defines an entry in the Mode Control Table used by LU6.2 services | 4 |
| DEFMSG | Defines delivery options for PPO messages | 4 |
| DEFOPSET | Defines an entry in the OSCT table | 4 |

| Command | Description | Auth |
|---|---|---|
| DEFTERM | (VOS3 only) Defines terminal type | 2 |
| DEFTRANS | Defines an entry in the TCT used by LU6.2 services | 4 |
| DELALIAS | Deletes an alias entry used by the Alias Name Translation Facility | 4 |
| DELCLASS | Deletes an NTS class definition | 4 |
| DELLINK | Deletes a dynamic link definition | 3 |
| DELLOGON | Deletes an entry form EASINET/MAI appl table | 4 |
| DELMODE | Deletes an APPC MCT table entry | 4 |
| DELOPSET | Deletes an APPC Option Set Control Table (OSCT) entry | 4 |
| DELTERM | (VOS3 only) Deletes terminal definition | 2 |
| DELTRANS | Deletes an APPC Transaction Control Table (TCT) entry | 4 |
| DISCONN | Disconnects either the current or the nominated processing region | 0/1 |
| DNR | Defines or modifies parameters for the Domain Name Resolver (DNR) | 4 |
| DOMAIN | Uniquely defines a domain | 3 |
| EDB | Defines, starts, stops, or deletes an EDB connection | 4 |
| END | Terminates a paused NCL process without further processing | 0/2 |
| ENTER | Defines the Enter key | 0 |
| EQUATE | Assigns a text value to a unique string | 0 |
| EVERY | Issues a command or series of commands at a specified time frequency | 0/2 |
| EXEC | Schedules an NCL process for serial execution | 0 |
| F | Issues a VTAM modify (F) command | 1 |

| Command | Description | Auth |
|---|---|---|
| FLUSH | Terminates an NCL process without further processing | 0/2 |
| FORCE | Forcibly inactivates a VTAM network node | 1 |
| FSPROC | Executes an NCL procedure in OCS full-screen mode | 0 |
| FSTOP | Forces an immediate shutdown of the system | 3 |
| FTS | Invokes File Transmission Services | 0 |
| FTSINIT | Modifies the status of FTS initiators | 0 |
| FTSMOD | Modifies the status of incoming or outgoing FTS transmissions | 0 |
| GO | Resumes processing for a paused NCL process | 0/2 |
| INACT | Inactivates a VTAM network node | 1 |
| INTQUE | Passes data to an NCL procedure's &INTREAD statement | 0/2 |
| ISR | Changes the status of ISR (Inter-System Routing) conversations | 2 |
| JOURNAL | Controls the journal data set | 2 |
| K | Clears the current OCS window | 0 |
| LIBPATH | Controls library path definitions | 4 |
| LIBRARY | Controls library definitions | 4 |
| LINK | Defines or changes the status of the INMC/NVS/APPC links | 3 |
| LINK START | Defines the attributes and then starts a link to a domain | 3 |
| LIST | Lists statements in a specific member of the procedure library | 0 |
| LOAD | Preloads copies of procedures, ASN.1 maps, or load modules into storage. | 4 |
| LOCK | Suspends current screen processing | 0 |
| LOG | Writes messages to the activity log | 0 |

| Command | Description | Auth |
|---|---|---|
| LOGSWAP | Swaps logging to the next available log | 3 |
| LSRPOOL | Sets attributes for the VSAM shared resource pool | 2 |
| LTITLE | Sets the title for your product region logo panel | 4 |
| LUTRACE | Traces session traffic to selected terminals | 4 |
| MAIDISC | Forces disconnection of an MAI-OC session | 0 |
| MAIINT | Interrupts an MAI-OC application | 0 |
| MAILOGON | Creates an MAI-OC session with another application | 0 |
| MAISEND | Sends data to an application connected by an MAI-OC session | 0 |
| MAISESS | Provides functions for the control of MAI/FS sessions | 0/4 |
| MAXUSERS | Displays and limits the maximum number of concurrent users | 3 |
| MSG | Sends a message to the specified OCS operators | 0 |
| NCLCHECK | Tests syntax for an NCL procedure without execution | 0 |
| NCLTEST | Sets, resets, or displays NCL test status for this window | 0 |
| NCLTRACE | Dynamically alters the trace status of an executing NCL process | 0/2 |
| NDB ALTER | Builds, rebuilds, or validates the key indexes for an NDB, or a field | 2 |
| NDB CREATE | Initializes a new NCL database | 2 |
| NDB FIELD | Allows a database field definition to be added, deleted, or updated | 2 |
| NDB PURGE | Frees up a locked or halted database | 2 |
| NDB RESET | Deletes all data from an NCL database | 2 |

| Command | Description | Auth |
|---|---|---|
| NDB START | Starts an NCL database | 2 |
| NDB STOP | Flags an NCL database as stopping | 2 |
| NDB UNLOAD | Unloads a copy of an NCL database | 2 |
| NETM | (VOS3 only) Passes a command to NETM for execution | 2 |
| NETMCNTL | (VOS3 only) Controls communication with NETM | 2 |
| NETSTAT | Invokes the TCP/IP NETSTAT command | T |
| NOTIFY | Sends a message to TSO/TSS users or CMS users | 0 |
| NOTRACE | Terminates a VTAM network trace | 1 |
| NPTAB | Changes the status of a user's NPF table | 0/4 |
| NRDRET | Restores all Non-Roll Delete messages for an OCS window | 0 |
| NSBRO | Creates and sends a general or specific broadcast | 3 |
| NSPCONN | Controls connections to CA NetSpy™ Network Performance address spaces | 4 |
| NSLOOKUP | Uses the name service to find the name or address of a host | T |
| NTSDBMOD | Alters database records or session keep counts on the NTS database | 4 |
| NTSMOD | Alters session trace and logging parameters of active sessions | 4 |
| OBEYFILE | Invokes the TCP/IP for MVS OBEYFILE command | T |
| OCSID | Sets or resets OCS window identifiers | 0 |
| OPNDST | Connects a specific terminal | 1/0 |
| OPSYS | Passes a command to the operating system for execution | 2 |
| ORDER | Reorders your OCS window display from top to bottom | 0 |

| Command | Description | Auth |
|---|---|---|
| PAGE | Ensures that output messages start at the top of the OCS window | 0 |
| PF | Displays and sets terminal Function keys | 0 |
| PING | Sends echo requests to a remote host | T |
| PPO | Starts and stops the VTAM PPO interface | 3 |
| PROFILE | Displays or modifies a user profile | 0/2 |
| PROFILE EDS | Enables or disables event notification | 0 |
| PURGE | Purges and deletes current timer-initiated commands or NCL locks | 0/2 |
| RECONN | Reconnects the current terminal session with a disconnected region | 0 |
| REPALIAS | Replaces an alias name entry used NEWS | 4 |
| REPCLASS | Replaces an existing SAW, RTM, Session, or Resource class definition | 4 |
| REPLINK | Replaces or defines a DLT definition | 3 |
| REPLOGON | Replaces an existing application logon entry | 4 |
| REPLY | Issues a VTAM REPLY command | 1 |
| REPMODE | Replaces or defines an APPC Mode Control Table (MCT) entry | 4 |
| REPOPSET | Replaces or defines an APPC Option Set Control Table (OSCT) entry | 4 |
| REPTRANS | Replaces or defines an APPC Transaction Control Table (TCT) entry | 4 |
| REQMS | Sends data across the CNM interface | 2 |
| RETURN | Exits from any mode or function and returns to the primary menu | 0 |
| ROUTE | Sends a command string to a remote system | 0 |
| SAWARE | Controls NTS activity | 4 |
| SCRIPT | Starts or flushes an MAI-FS script procedure | 0 |

| Command | Description | Auth |
|---|---|---|
| SECUSER | Issues a command to a VS service machine | 2 |
| SHOW AIF | Displays AIF users | 0 |
| SHOW ALLOC | Displays data sets allocated by the ALLOC command | 0 |
| SHOW AOMABEND | Displays diagnostic information if AOM abends | 0 |
| SHOW AOMSTAT | Displays AOM statistics | 0 |
| SHOW APPC | Displays the status of all or selected APPC conversations | 0 |
| SHOW ATF | Displays information about the ATF tables | 0 |
| SHOW BRO | Displays the current general broadcast messages | 0 |
| SHOW BUFF | Displays the current system storage and buffer use | 0 |
| SHOW CNMTRACE | Shows active CNM trace requests | 0 |
| SHOW COMMANDS | Displays commands available to user or system | 0 |
| SHOW COMP | Displays 3270 data stream compression statistics | 0 |
| SHOW CONSOLES | Displays the consoles currently allocated for use by AOM | 0 |
| SHOW DEBUG | Displays NCL debug sessions | 0/2 |
| SHOW DEFALIAS | Displays one or more DEFALIAS entries | 0 |
| SHOW DEFCLASS | Displays NTS class definitions | 0 |
| SHOW DEFLINK | Displays current dynamic link definitions | 0 |
| SHOW DEFLOGON | Displays the current entries in the DEFLOGON table | 0 |
| SHOW DEFMODE | Displays APPC Mode Control Table (MCT) entries | 0 |
| SHOW DEFMSG | Displays the current DEFMSG delivery options | 0 |

| Command | Description | Auth |
|---|---|---|
| SHOW DEFOPSET | Displays APPC Option Set Control Table (OSCT) entries | 0 |
| SHOW DEFTERM | (VOS3 only) Displays terminal definitions | 0 |
| SHOW DEFTRANS | Displays APPC Transaction Control Table (TCT) entries | 0 |
| SHOW DNR | Displays parameters or statistics for the Domain Name Resolver (DNR) | 0 |
| SHOW DOMAINS | Displays the domain information of connected systems | 0 |
| SHOW EDB | Displays information about external database connections | 0 |
| SHOW EDBSTAT | Displays statistics about currently defined EDB connections | 0 |
| SHOW EDBUSER | Displays information about EDB users | 0 |
| SHOW EDBUSTAT | Displays statistics about EDB users | 0 |
| SHOW EDS | Displays current Event Distribution Services PROFILE definitions | 0 |
| SHOW EPS | Displays EndPoint Services information | 0 |
| SHOW EQUATES | Displays the current EQUATE strings available to this user | 0 |
| SHOW EXEC | (OS/VS only) Displays the names of members in a procedure library | 0 |
| SHOW FTS | Displays the status of file transmissions | 0 |
| SHOW ISR | Displays ISR status information | 0 |
| SHOW ISRSTATS | Displays ISR statistics | 0 |
| SHOW LIB | Displays libraries which have been defined to the system | 0 |
| SHOW LICENSE | Displays the active licensing details and expiry dates | 0 |
| SHOW LINK | Displays the status of INMC or APPC links | 0 |
| SHOW LOCKS | Displays the status of all locks held within the system | 0 |

| Command | Description | Auth |
|---|---|---|
| SHOW LOGS | Displays the status of the system activity logs | 0 |
| SHOW LSR | Displays LSR status information | 0 |
| SHOW LUTRACE | Displays the status of all LUTRACE requests | 0 |
| SHOW MAI | Displays information about MAI sessions for this user or all users | 0 |
| SHOW MAISTAT | Displays information about the status of MAI subtasks | 0 |
| SHOW MAP | Displays information about defined Mapping Services maps | 0 |
| SHOW MSGQ | Displays message queue depths | 0 |
| SHOW NCL | Displays the status of active NCL processes | 0/2 |
| SHOW NCLGLBL | Displays the names of any defined NCL global variables | 2 |
| SHOW NCLLOCKS | Displays the status of all NCL locks held within the system | 0 |
| SHOW NCLSTAT | Displays the status of NCL procedures currently in storage | 0 |
| SHOW NCLVARS | Displays information about the variables in use by NCL procedures | 0/4 |
| SHOW NDB | Displays information about currently active or halted NDBs | 0 |
| SHOW NDBUSER | Displays a list of all users currently signed on to NCL databases | 0 |
| SHOW NETMCNTL | (VOS3 only) Displays the status of NETM connectivity | 0 |
| SHOW NPF | Displays user’s Network Partitioning Facility (NPF) tables | 0 |
| SHOW NPTAB | Displays the status of NPF resource tables | 0 |
| SHOW NRD | Displays the current queue of Non-Roll Delete messages | 0 |

| Command | Description | Auth |
|---|---|---|
| SHOW NSPCONN | Displays the status of connections to CA NetSpy Network Performance address spaces | 0 |
| SHOW NTS | Displays NTS resource or session information | 0 |
| SHOW NTSDBMOD | Displays currently executing NTSDBMOD & SHOW SKEEP processes | 0 |
| SHOW NTSSTATS | Displays NTS statistics | 0 |
| SHOW NTSUSER | Displays NTS users | 0 |
| SHOW OCS | Displays information about OCS users | 0 |
| SHOW PANELS | Displays panel queue information | 0 |
| SHOW PARM | Displays startup parameters | 0 |
| SHOW PATH | Displays current panel library path definitions | 0 |
| SHOW PAUSE | Displays the current PAUSE or WAIT status | 0/2 |
| SHOW PPIUSERS | Displays PPI user statistics | 0 |
| SHOW PPOSTAT | Displays PPO/SPO statistics | 0 |
| SHOW REPLY | Displays VTAM messages that require a reply | 0 |
| SHOW SCNT | Displays the current session count | 0 |
| SHOW SERVER | Displays the status of registered server processes | 0 |
| SHOW SESS | Displays the terminals that are in session | 0 |
| SHOW SKEEP | Displays the NTS session keep counts for historical sessions | 0 |
| SHOW SNAMS | Displays SNA information | 0 |
| SHOW SOCKETS | Displays information about the use of TCP/IP services | T |
| SHOW SSISTATS | Displays Subsystem Interface (SSI) statistics | 0 |
| SHOW SSIUSERS | Displays information about SSI users | 0 |

| Command | Description | Auth |
|---|---|---|
| SHOW STRACE | Displays NTS session trace activity | 0 |
| SHOW SUBSYS | Displays user subsystem status | 0 |
| SHOW SYSCONS | Displays a list of currently logged on operating system consoles | 0 |
| SHOW SYSPARMS | Displays the current SYSPARMS settings | 0 |
| SHOW TCPIP | Displays information about the use of TCP/IP services | T |
| SHOW TERM | Displays the terminals that are in session | 0 |
| SHOW TIMER | Displays current timer initiated commands | 0 |
| SHOW TIMEZONE | Displays current time zones and the system time zone offset | 0 |
| SHOW TSO | Displays current TSO users | 0 |
| SHOW TSS | Displays current TSS users | 0 |
| SHOW UDB | Displays VSAM data set information | 0 |
| SHOW UDBUSER | Displays NCL UDB user information | 0 |
| SHOW USERS | Displays the current signed on users | 0 |
| SHOW VARTABLES | Displays information about VARTABLES | 0 |
| SHOW VMOP | Displays VMOPERATOR active sessions | 0 |
| SHOW VSAM | Displays extended VSAM data set information | 0 |
| SHOW XMIT | Displays the status of all or selected FTS Transmission Requests | 0 |
| SHOW XNFTRACE | Displays the status of all XNFTRACE requests | 0 |
| SHUTDOWN | Commences or cancels an orderly shutdown of the system | 3 |
| SIGNOFF | Signs off from a remote system | 0 |
| SIGNON | Signs on to a remote system | 0/2 |
| SPLIT | Opens or adjusts size of an OCS window | 0 |

| Command | Description | Auth |
|---|---|---|
| SPO | Starts and stops the VTAM Secondary Program Operator interface | 3 |
| SSI | Signs off or stops Subsystem Interface (SSI) | 0/3 |
| START | Starts executing an asynchronous NCL process | 0 |
| STATUS | Displays current general system status | 0 |
| STRACE | Starts and stops session trace activity | 4 |
| SUBMIT | Passes a command to a background environment for processing | 0/2 |
| SUBSYS | Defines, starts, stops, and deletes subsystems | 3 |
| SUSLOGON | Suspends an entry or entries in the DEFLOGON table | 3 |
| SWAP | Swaps current logical windows | 0 |
| SYNCTIME | Synchronizes the date and time of the region with the local system | 2 |
| SYSCMD | Sends a command to the operating system | 2 |
| SYSLOG | Issues VMOPERATOR log browse commands | 2 |
| SYSMON | Sends data to System Monitor in 3600/4700 | 1 |
| SYSPARMS | Defines or changes system default values.<br>**Note:** For more information, see the *Administration Guide.* | 4/9 |
| TERMINAL | Provides extended color and/or highlighting data streams to terminals | 0 |
| TCPIP MODIFY | Alters the current trace options setting | 4 |
| TCPIP QUIESCE | Stops the interface when all the sockets are closed | 4 |
| TCPIP START | Initiates TCP/IP services | 4 |
| TCPIP STOP | Terminates TCP/IP services | 4 |

| Command | Description | Auth |
|---|---|---|
| TELNET | Starts a Telnet connection | T |
| TIME | Displays the current date and time | 0 |
| TIMEZONE | Maintains time zone names and offsets | 3 |
| TITLE | Sets the title to display on the top line of the OCS window | 4 |
| TNCMD | Sends a Telnet command on a connection to a remote host | T |
| TNDISC | Disconnects a Telnet connection | T |
| TNSEND | Sends data on a Telnet connection | T |
| TRACE | Initiates a VTAM network trace | 1 |
| TRACEROUTE | Traces the route taken by TCP/IP packets to a remote host | T |
| TRANSMIT | Requests a data set transmission | 0 |
| UDBCTL | Controls the status of User DataBases (UDBs) | 3/4 |
| UNALLOC | Deallocates a closed data set | 0/3 |
| UNLOAD | Unloads copies of procedures, ASN.1 maps, load modules, or panels from storage. | 4 |
| V | Issues a VTAM VARY command | 1 |
| X | Exits from OCS screen mode | 0 |
| XLATE | Tests alias name translation | 1 |
| XNF | Stops all XNF connections | 3 |
| XNFTRACE | Initiates or terminates tracing of XNF connectivity | 4 |

# Index